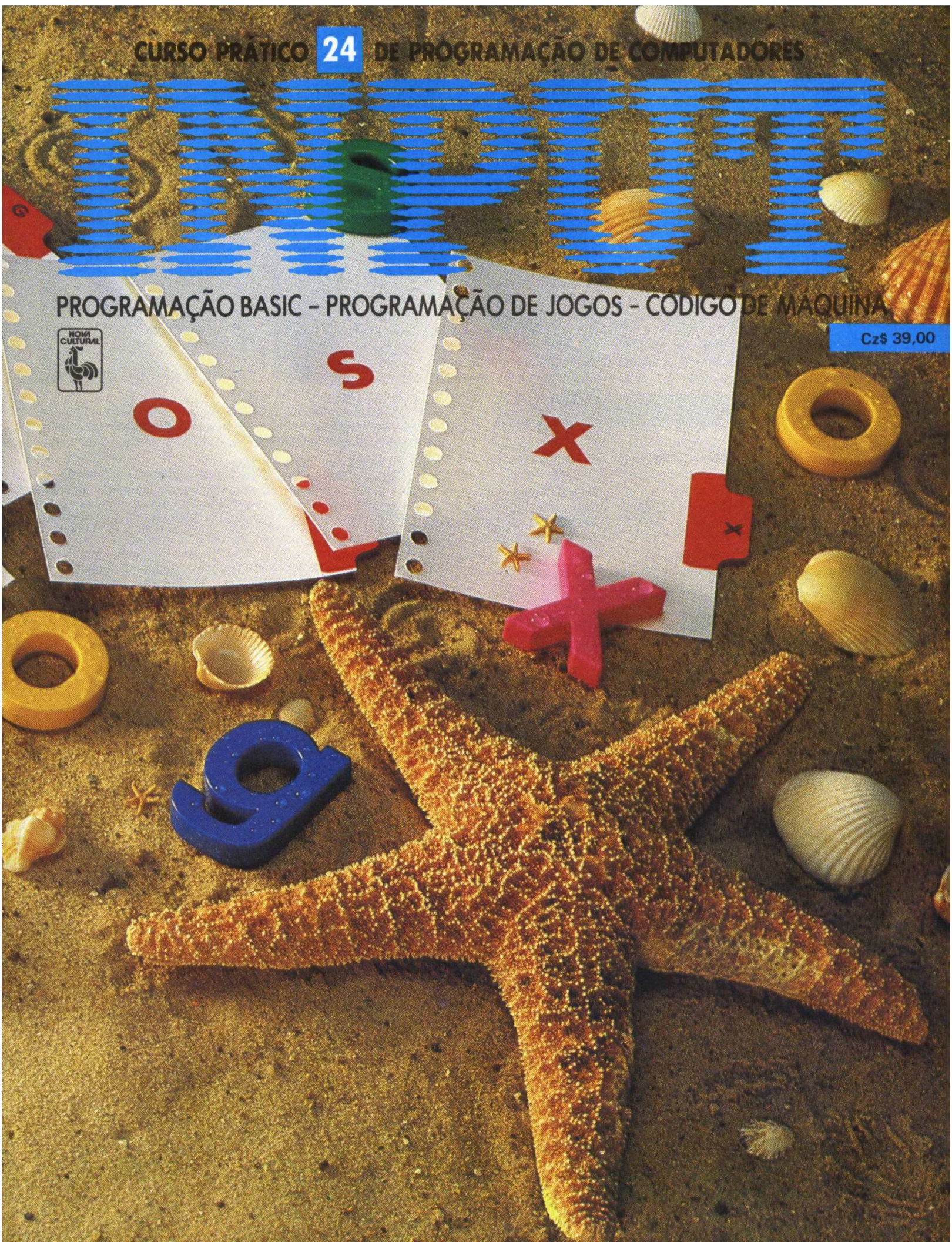
CURSO PRÁTICO 24 DE PROGRAMAÇÃO DE COMPUTADORES

DIPLOM

PROGRAMAÇÃO BASIC – PROGRAMAÇÃO DE JOGOS – CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 39,00



INPUT

Vol. 2 Nº 24

NESTE NÚMERO

PROGRAMAÇÃO DE JOGOS

AS REGRAS DO JOGO

Vinte-e-um: as regras do jogo normal e da versão computadorizada. Funcionamento do programa. Como programar decisões. Perder ou ganhar. 461

PROGRAMAÇÃO BASIC

ROTINAS DE ORDENAÇÃO

O que é ordenação. Ordenação tipo bolha. Vantagens da rotina de ordenação de busca binária. As rotinas de Shell e Shell-Metzner..... 468

CÓDIGO DE MÁQUINA

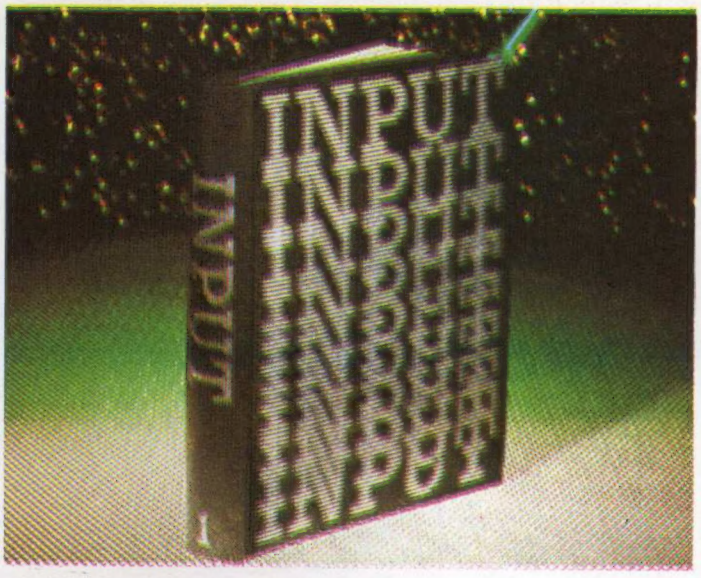
DRAGÃO ANIMADO

Como colocar a figura no quadriculado. Movimentos na tela. Faça o dragão cuspir fogo 474

PROGRAMAÇÃO BASIC

ANIMAÇÃO GRÁFICA NO TRS-COLOR

Animação de um caractere definido pelo usuário Os comandos **GET** e **PUT**. Como dimensionar a matriz. Técnicas de animação 478



PLANO DA OBRA
INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...
NÃO FIQUE COM A COLEÇÃO INCOMPLETA
Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em *São Paulo,* os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André. No *Rio de Janeiro,* av. Mem de Sá, 191/193, Centro.
2. **Por carta** — Envie para:
   DINAP — Distribuidora Nacional de Publicações
   Números Atrasados
   Estrada Velha de Osasco, 132 — Jardim Teresa
   CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Varais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre o título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO
Encaminhe seus comentários, críticas, sugestões ou reclamações ao
SERVIÇO DE ATENDIMENTO AO LEITOR
Caixa Postal 9 442, São Paulo — SP.

EDITOR
RICHARD CIVITA
NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues,
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shozi Ikeda,
Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Antonio José Filho, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena,
Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

Secretária de Redação/Coordenadora: Stefania Crema
Secretário de Redação: Mauro de Queiroz

Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas-SP)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Joaquim Celestino da Silva
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado:
Wagner M. P. Nabuco de Araújo

CLC

A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer e Cultura
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez, Menahen M. Politi, Renê C. X. Santos, Stélio Alves Campos

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986; 2ª edição, 1987.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta pela AM Produções Gráficas Ltda. e impressa pela Companhia Lithographica Ypiranga.

AS REGRAS DO JOGO

- AS REGRAS DO JOGO NORMAL E DA VERSÃO COMPUTADORIZADA
- FUNCIONAMENTO DO PROGRAMA
- AS DECISÕES DA BANCA
- PERDER OU GANHAR

Para completar nosso jogo de vinte-e-um, só falta ensinar o computador a fazer as vezes da banca. É hora, portanto, de se preparar para enfrentá-lo: conheça as regras do jogo.

Antes de passar à última seção do programa, convém conhecer as regras do jogo.

O vinte-e-um utiliza um baralho com 52 cartas. As cartas que vão do 2 ao 10 têm valor igual ao seu número. As figuras valem 10 pontos e o ás pode valer 1 ou 11, conforme a decisão do jogador. Em nosso jogo, o computador calcula os pontos automaticamente.

Joga-se, em geral, com dinheiro ou fichas. Nosso computador é programado para marcar os pontos com fichas. Cabe a ele desempenhar o papel da banca, sendo sempre quem dá as cartas.

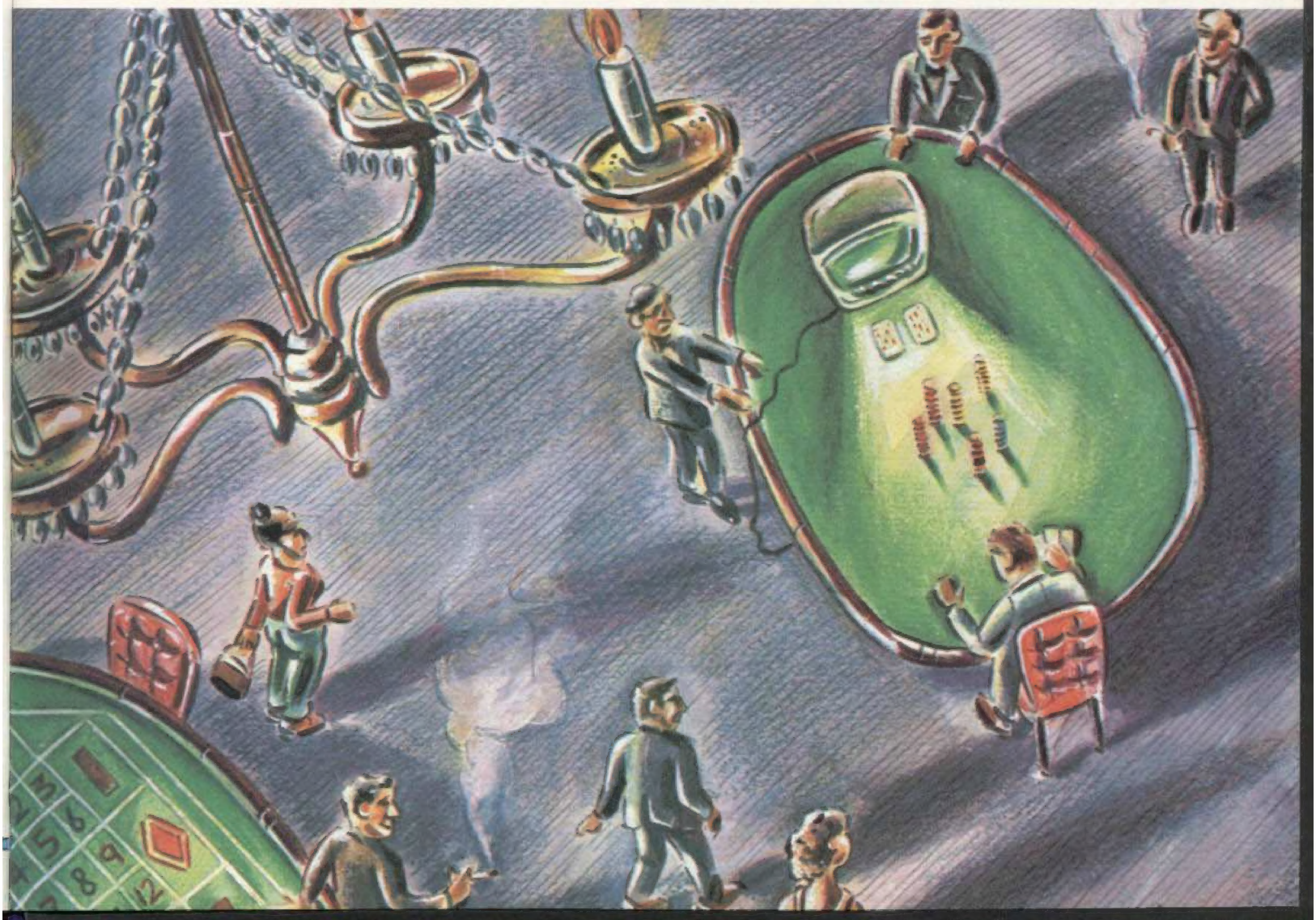
No início do jogo, o computador embaralha as cartas e distribui duas, viradas para baixo ("fechadas"). A carta do jogador aparece "aberta" na tela, mas o programa foi feito de modo que o computador não conheça as cartas do jogador. Este faz sua aposta, antes que ele e a banca recebam outra carta.

O objetivo do jogo é ter uma mão melhor que a da banca, isto é, um maior número de pontos na soma das cartas. Quem tiver um total maior que 21, estourou, perdendo a aposta. Com um total entre 16 e 21, o jogador só ganha da banca se esta possuir um total menor, ou se tiver estourado. Assim, a banca tem sempre a vantagem do empate.

Existem duas mãos especiais — o "natural": um ás e um dez ou um ás e uma figura, somando 21 em duas cartas; e a "mão de cinco": uma mão com cinco cartas, cujo total é igual ou inferior a 21. Um natural do jogador ganha de qualquer mão da banca, exceto de outro natural. Uma mão de cinco também ganha de qualquer coisa, menos de um natural ou outra mão de cinco.

COMO FUNCIONA O PROGRAMA

Depois que o jogador recebeu a segunda carta, o programa verifica se ele



conseguiu um natural. Se não conseguiu, mas obteve um total maior ou igual a 16, e está satisfeito com este valor, ele pode "parar", não recebendo mais cartas. A vez de jogar passa, então, à banca. Se obteve um total inferior a 16, ou não está satisfeito com o valor de sua mão, pode receber mais cartas pelo ato de "pedir" ou de "comprar". No jogo normal, quando se compra uma carta, a aposta original é dobrada e a banca dá a carta fechada. Na nossa versão computadorizada, não há diferença entre cartas fechadas ou abertas, e pedir cartas não dobra a aposta, pois todas são dadas abertas. O jogador não poderá comprar depois de ter pedido alguma carta e, ao chegar na quinta carta, só poderá pedir. Após cada carta fornecida, o computador verifica se o jogador ou a banca ultrapassaram os 21 pontos.

Quando o jogador estoura, a banca ganha a aposta e não precisa jogar. Quando ele tem 21 ou menos, a banca mostra suas duas cartas e decide se pede mais cartas ou pára por ali. Se as duas cartas da banca são um natural, ela ganha a aposta. Caso contrário, a banca pede mais cartas até ficar satisfeita com o total, conseguir uma mão de cinco ou então estourar — para alegria do jogador.

Se a banca pára com um total menor que 21 e sem ter obtido uma mão de cinco, surge na tela a mensagem: a BANCA PAGA... totais que tenham um ponto a mais que o valor de sua mão. Se a mão da banca for 21, a mensagem NATURAIS E MÃOS DE CINCO APENAS é mostrada. E, caso obtenha uma mão de cinco, o jogador é informado: APENAS NATURAIS. No jogo convencional, cada jogador declararia o valor de sua mão, se tivesse vencido a banca. Na nossa versão, o computador soma as cartas do jogador para saber seu total. Se o jogador tiver um total maior ou igual ao que a banca paga, ele vence; se tiver menos, perde. Quando vence, a banca paga um valor igual a sua aposta, ou seja, ele recebe o dobro do que apostou. Nenhuma ficha extra é paga a naturais ou mãos de cinco, e não é permitido blefar.

O jogador também não pode dar as cartas nem fazer papel de banca — no jogo convencional a banca pertence a quem consegue um natural. Resta, assim, ao jogador, a difícil tarefa de quebrar a banca, o que ocorre quando ele consegue acumular mais de 1 000 fichas. No início da partida, o jogador recebe 100 fichas e, caso perca todas, perde também o jogo.

S

Carregue as duas seções iniciais do programa e acrescente as linhas seguintes para completar seu jogo.

```
2500 PAUSE 50
```

```
2510 LET DPF=B: LET AF=B: LET X
=C: LET Y=10
2520 FOR J=C TO 2
2530 LET Z=O(J): GOSUB 5500
2540 IF VA>10 THEN  LET VA=10
2545 FOR K=C TO 2: LET W(K)=W(K
)+VA: NEXT K
2546 IF VA=C AND AF=B THEN  LET
 W(2)=W(2)+10: LET AF=C
2550 LET X=X+6: NEXT J
2560 IF W(2)=21 THEN  PRINT  PA
PER 2; INK 7;AT 14,18;" NATURAL
! ";AT 15,16;" A BANCA GANHA ":
 LET DPF=C: GOTO 2000
2600 IF PF=C THEN  GOTO 2720
2610 PAUSE 75
2630 IF W(C)>21 THEN  PRINT  PA
PER C; INK 7;AT 21,B;" A BANCA
ESTOURA! ";: GOTO 2740
2635 IF W(2)>21 THEN  LET W(2)=
W(C)
2640 IF W(2)<16 THEN  GOTO 2800
2650 IF FF=C THEN  GOTO 2800
2660 LET PR=(W(2)-8)/13
2670 IF PR<RND THEN  GOTO 2800
2700 IF W(2)>21 THEN  LET W(2)=
W(C)
2710 IF W(2)=21 THEN  PRINT AT
20,B;" A BANCA PAGA SOMENTE NAT
URAIS   E MAOS DE CINCO": GOTO
2000
2720 PRINT  PAPER 2;AT 21,B;" A
 BANCA OBTEVE ";W(2)+C;
2725 IF S(2)>21 THEN  LET S(2)=
S(C)
2730 IF W(2)>=S(2) THEN  PRINT
 PAPER 2;" E GANHOU! ": GOTO 20
00
2740 PRINT  PAPER 2;" VOCE GANH
OU ": LET CP=CP+BET*2: GOTO 200
0
2800 LET Z=C(CC): GOSUB 5500
2805 IF VA>10 THEN  LET VA=10
2810 FOR K=C TO 2: LET W(K)=W(K
)+VA: NEXT K
2820 IF VA=C AND AF=B THEN  LET
 W(2)=W(2)+10: LET AF=C
2822 IF W(1)<22 AND X=25 THEN
PRINT  PAPER 2; INK 7;AT 21,B;"
 MAO DE CINCO! ": GOTO 2000
2825 GOSUB 7000
2830 LET X=X+6: GOTO 2610
```

A linha 2500 provoca um segundo de pausa e, então, o programa inicia a jogada da banca. A 2510 coloca zero no sinalizador de naturais e no sinalizador de ases da banca, e acerta as coordenadas da primeira carta.

DISTRIBUIÇÃO DAS CARTAS

As duas cartas iniciais são "viradas" pelo **FOR...NEXT** entre as linhas 2520 e 2550; a linha 2530 mostra as cartas. A 2540 atribui o valor dez às cartas de figuras, antes que a 2545 some seu valor com o total da banca (dois totais, se houver um ás). A 2546 soma dez a **W(2)**, se uma das cartas for um ás, e coloca 1 no indicador de ases. A linha acerta a posição da próxima carta.

A linha 2560 verifica se foi obtido um natural, anunciando a vitória da banca, se for o caso. O indicador de naturais da banca passa a valer 1. Se o jogador também tem um natural — linha 2600 —, o programa vai à rotina que cuida das mensagens: "quem tem o que" e "quem venceu". Após a pausa da linha 2610, a linha 2630 verifica se a banca estourou; em caso afirmativo, surge a mensagem A BANCA ESTOUROU. A linha 2635 verifica se o maior dos totais — quando há um ás — excedeu 21. O programa só considera o menor total. Se a banca tiver menos de 16 pontos, a linha 2800 vira outra carta.

QUER MAIS CARTAS?

As linhas 2660 e 2670 decidem se a banca quer outra carta. Não há regra fixa para isso. Não vale a pena impor um limite acima do qual a banca não pede mais cartas, pois ela se tornará previsível e o jogador logo saberá como derrotá-la. É necessário introduzir um fator de incerteza (como fazemos aqui), que torne impossível prever quando a banca vai parar de pedir cartas.

Para isso, o programa deverá fazer o computador se comportar de maneira semelhante a um ser humano. Coloque-se no lugar da banca: tendo um total de 20 pontos, você se sentiria menos inclinado a pedir outra carta do que se tivesse 16. Quando o computador compara PR com um número randômico, introduz-se um fator de incerteza e, ao mesmo tempo, a decisão passa a depender do valor da mão da banca.

O RESULTADO FINAL

A linha 2700 troca o valor de **W(2)** pelo de **W(1)**, se **W(2)** contiver mais de 21. Se a banca conseguir 21 pontos, a linha 2710 imprime na tela a mensagem A BANCA PAGA SOMENTE NATURAIS E MÃOS DE CINCO. Caso possua uma mão inferior a 21, a linha 2720 escreve A BANCA OBTEVE, seguido de um valor um ponto mais alto que o total alcançado pela banca.

A linha 2730 compara o total do jogador com o da banca. Quando esta vence, a linha 2730 informa o fato; caso contrário, a linha 2740 anuncia a vitória do jogador e calcula o novo total de fichas.

A parte final do programa — linhas 2800 a 2830 — cuida da distribuição das cartas da banca.

A linha 2805 atribui dez pontos às cartas de figura. O valor da carta é somado ao de **W(1)** e **W(2)** na linha 2810. Se houver um ás, à linha 2820 cabe executar as alterações necessárias antes que a linha 2822 verifique se se conseguiu uma mão de cinco. A linha 2825 ajusta o monte de cartas, preparando uma nova distribuição. A linha 2830 calcula a posição onde a carta vai ser desenhada.

Apague o **GOTO 86** do final da linha 650 e adicione as linhas seguintes:

```
500 GOSUB 4000:GOTO 540
510 CX=CX+32:GOSUB 3500:NC=NC+1
530 IF DL>21 THEN 620
540 DT=DL+10*(DA AND(DL<12)):IF
 DT=21 AND NC=2 THEN GOSUB 5000
:PRINT#1,"A BANCA TEM UM NATURA
L":GOSUB 5500:GOTO 610
550 IF NC=5 THEN GOSUB 5000:PRI
```

```
NT#1,"A BANCA TEM UMA MÃO DE CI
NCO":GOSUB 5500:GOTO 610
560 R=20-DT:IF RND(1)*100<R*R*R
 OR DT<16 OR PS=1 THEN 510
570 GOSUB 5000:IF P5=1 THEN PRI
NT#1,"VOCÊ TEM UMA MÃO DE CINCO
" ELSE PRINT#1,"Você tem";PT
580 GOSUB 5500:GOSUB 5000:PRESE
T(8,80):PRINT#1,"A banca paga s
omente";:PRESET(8,92):IF DT<21
THEN PRINT#1,"totais maiores ou
 iguais a";DT+1 ELSE PRINT#1,"n
aturais e mãos de cinco"
590 GOSUB 5500:IF P5=1 THEN 630
600 IF DT<PT THEN 630
610 GOSUB 5000:PRINT#1,"A banca
 venceu":GOSUB 5500:GOTO 690
620 GOSUB 5000:PRINT#1,"A banca
 estourou. Você venceu":GOSUB 5
500:GOTO 640
630 GOSUB 5000:PRINT#1,"Você ve
nceu":GOSUB 5500
640 MN=MN+2*BT:GOSUB 5000:GOSUB
 7000:GOTO 700
650 GOSUB 5000:PRINT#1,"VOCÊ TE
M UM NATURAL":PF=1:GOSUB 5500
660 GOSUB 4000:IF DL=11 THEN GO
SUB 5000:PRINT#1,"MAS A BANCA T
EM UM TAMBÉM":GOSUB 5500:GOTO 6
10
670 GOTO 630
690 IF MN<1 THEN GOSUB 5000:PRI
NT#1,"Suas fichas acabaram":GOS
UB 5500:GOTO 790
700 IF PF=1 THEN GOSUB 5000:PRI
NT#1,"Embaralhando as cartas":G
OSUB 5500:GOSUB 1500:GOTO 750
710 NF=N-1:IF NA>N THEN NF=N+51
720 FOR X=NF TO NA+1 STEP -1:Q=
INT((X-NA)+NA):T=SQ(X):SQ(X)=SQ
(Q):SQ(Q)=T:NEXT:IF NA>N THEN 7
40
730 FOR X=0 TO 9:SQ(X+52)=SQ(X)
:NEXT:GOTO 750
740 FOR X=0 TO 9:SQ(X)=SQ(X+52)
:NEXT
790 GOSUB 5000:PRINT#1,"Quer jo
gar novamente (S/N)?":CLOSE1
800 A$=INKEY$:IF A$<>"N" AND A$
<>"N" THEN 800
810 IF A$="S" THEN RUN
820 COLOR 15,4,4:END
3500 GOSUB 1000:GOSUB 2000:IF N
M=1 THEN DA=1
3510 IF NM>10 THEN DL=DL+10 ELS
E DL=DL+NM
3520 RETURN
4000 NN=N:N=D1:CX=31:CY=105:GOS
UB 3500
4010 N=D2:CX=63:GOSUB 3500:N=NN
4020 NC=2:RETURN
```

Antes da banca jogar, vêem-se na tela a mão do jogador e as duas cartas da banca fechadas. A banca abre, então, suas cartas — a linha 500 chama a sub-rotina 4000. Esta abre a primeira carta, chamando inicialmente outra sub-rotina, da linha 3500, que cuida do desenho das cartas e calcula os pontos da banca. Na linha 4010 a segunda carta é aberta, chamando-se a sub-rotina novamente, depois que a posição horizontal da carta foi modificada. O número da carta é atualizado e a sub-rotina termina na linha 4020.

A linha 510 muda e acerta a posição horizontal da carta seguinte e chama a sub-rotina da linha 3500. Assim, a banca recebe uma nova carta. A linha 530 verifica se a banca ultrapassou 21 pontos; a linha 620 avisa o jogador, se for o caso. Se a banca conseguiu um natural, o indicador de naturais da banca passa a valer 1 e o jogador recebe uma mensagem. A linha 550 detecta mãos de cinco.

AS DECISÕES DA BANCA

A parte mais interessante do programa está na linha 560, que faz o computador decidir se quer ou não mais cartas. Seria mais fácil que a máquina parasse de pedir cartas quando atingisse um determinado total — 19, por exemplo. Porém, seria igualmente fácil para o jogador derrotar a banca, uma vez que soubesse do fato. É necessário, portanto, introduzir um fator de incerteza na decisão do computador, tornando-a imprevisível. Isso não significa que se deva programar uma estratégia suicida — fazendo, por exemplo, com que a banca habitualmente peça mais cartas, já tendo 20 pontos.

Criamos, então, uma variável **R**, igual a 20, menos os pontos da banca. A seguir, um número qualquer entre 0 e 100 é escolhido ao acaso e comparado com **R*R*R**. Se o número for menor, se a banca tiver menos de 16 pontos ou se o jogador conseguir uma mão de cinco, o computador pede mais uma carta. O modo como calculamos **R** faz a chance da banca pedir mais cartas diminuir à medida que seu total de pontos aumenta. Raramente ela pedirá mais cartas já tendo 19 pontos, mas quase sempre o fará quando ainda tiver 16.

Usando o mesmo raciocínio e modificando um pouco a linha 560, podemos, além de variar a dificuldade do jogo, fazer com que a banca jogue baseando-se também nas cartas abertas do jogador, por exemplo.

O RESULTADO FINAL

A linha 570 mostra o valor da mão do jogador, apontando a ocorrência de uma mão de cinco, se for o caso. A linha 580 indica as mãos que vencem a banca. Se o jogador tiver uma mão de cinco e a banca não, a linha 590 exibe a mensagem VOCÊ VENCEU. De modo semelhante, a linha 600 compara os totais do jogador e da banca, informando o resultado através da linha 610 ou da linha 630.

Quando o jogador vence, a linha 640 acrescenta as fichas ganhas às que o jogador já tinha. Sempre que alguém consegue um natural, as cartas são embaralhadas. A linha 700 verifica o indicador de naturais e as linhas 710 a 740 embaralham as cartas.

Após o anúncio de que o jogador conseguiu um natural, a linha 660 verifica se a banca obteve o mesmo tipo de mão. Em caso afirmativo, vence a banca e a linha 610 indica o resultado. Se o jogador ganhar, caberá à linha 670 informar o fato.

Quando se acabam as fichas do jogador, a linha 690 diz VOCÊ PERDEU TODAS AS FICHAS.

Finalmente, uma pequena rotina (linhas 790 a 820) oferece ao jogador a opção de jogar novamente.

Apague o **GOTO 190** do fim da linha 650 e acrescente as linhas seguintes.

```
500  GOSUB 4000: GOTO 540
510 CX = CX + 54: GOSUB 3500:NC
 = NC + 1
520  FOR K = 1 TO 1700: NEXT K
530  IF DL > 21 THEN 620
540 DT = DL + 10 * (DA AND (DL
< 12)): IF DT = 21 AND NC = 2 T
HEN  TEXT : HOME : PRINT "A BAN
CA TEM UM NATURAL":PF = 1: GOTO
 610
550  IF NC = 5 THEN  TEXT : HOM
E : PRINT "A BANCA TEM UMA MAO
DE CINCO": GOTO 610
560 R = 20 - DT: IF  RND (1) *
100 < R * R * R OR DT < 16 OR P
5 = 1 THEN 510
570  TEXT : HOME : IF P5 = 1 TH
EN  PRINT "VOCE TEM UMA MAO DE
CINCO": GOTO 580
575  PRINT : PRINT "VOCE TEM ";
PT
580  PRINT : PRINT "A BANCA PAG
A ";: IF DT < 21 THEN  PRINT "V
ALORES MAIORES OU IGUAIS A ";DT
 + 1: GOTO 590
585  PRINT "NATURAIS E MAOS DE
CINCO APENAS"
590  FOR K = 1 TO 500: NEXT : I
F P5 = 1 THEN 630
600  IF DT < PT THEN 630
610  PRINT : PRINT "A BANCA GAN
HOU": GOTO 690
620  TEXT : HOME : PRINT "A BAN
CA ESTOUROU. VOCE GANHOU": GOTO
 640
630  PRINT : PRINT "VOCE GANHOU
"
640 MN = MN + 2 * BT: GOTO 700
660  GOSUB 4000: HOME : TEXT :
IF DL = 11 THEN  PRINT "MAS A B
ANCA TEM A MESMA COISA": GOTO 6
10
670  GOTO 630
690  PRINT : IF MN < 1 THEN  PR
INT "VOCE PERDEU TODAS AS FICHA
S": GOTO 790
700  IF PF = 1 THEN  PRINT : PR
INT "EMBARALHANDO AS CARTAS": G
OSUB 1500: GOTO 750
710 NF = N - 1: IF NA > N THEN
NF = N + 51
720  FOR X = NF TO NA + 1 STEP
 - 1:Q =  INT ( RND (1) * (X -
NA)) + NA:T = SQ(X):SQ(X) = SQ(
Q):SQ(Q) = T: NEXT : IF NA > N
THEN 740
730  FOR X = 0 TO 9:SQ(X + 52)
= SQ(X): NEXT : GOTO 750
740  FOR X = 0 TO 9:SQ(X) = SQ(
X + 52): NEXT
790  PRINT : PRINT "QUER JOGAR
NOVAMENTE (S/N) ?"
800  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 800
810  IF A$ = "S" THEN 150
820  END
3500  POKE  - 16299,0: POKE  -
16304,0: GOSUB 1000: GOSUB 2000
: IF NM = 1 THEN DA = 1
3510  IF NM > 10 THEN DL = DL +
 10: GOTO 3520
3515 DL = DL + NM
3520  RETURN
4000 NN = N:N = D1:CX = 6:CY =
100: GOSUB 3500
```

```
4010 N = D2:CX = 60: GOSUB 3500
:N = NN
4020  FOR K = 1 TO 3500: NEXT :
NC = 2: RETURN
```

Como o programa completo é muito extenso, utilizamos a página gráfica 2. Nunca use a página 1, pois o programa será danificado.

Quando a vez de jogar passa à banca, as cartas do jogador e as duas cartas fechadas do adversário aparecem na tela. A banca, em primeiro lugar, abre suas cartas — a linha 500 chama a sub-rotina 4000, que abre a primeira carta da banca chamando outra sub-rotina, da linha 3500. Esta sub-rotina liga o modo gráfico da página 2, sem apagar seu conteúdo, desenha a carta e calcula o total de pontos. A linha 4010 abre a segunda carta, que chama a mesma sub-rotina após ter modificado a posição horizontal da carta. O número da carta é atualizado e a sub-rotina termina na linha 4020.

A linha 510 calcula a posição horizontal da nova carta e volta a chamar a sub-rotina 3500. Assim, a banca recebe mais uma carta. A linha 520 provoca uma pausa antes que a linha 530 verifique se a banca não ultrapassou os 21 pontos — a linha 630 informa o fato, se necessário. A linha 540 verifica se a banca obteve um natural; em caso afirmativo, coloca 1 no indicador de naturais e exibe uma mensagem na tela. A linha 550 detecta uma mão de cinco.

AS DECISÕES DA BANCA

A parte mais interessante do programa está na linha 560, que faz o computador decidir se quer ou não mais cartas. Seria mais fácil programar a máquina de modo que parasse de pedir cartas sempre que atingisse um determinado valor — 19, por exemplo. Mas seria igualmente fácil para o jogador derrotá-la, uma vez que tivesse conhecimento dessa estratégia inflexível. É necessário introduzir no programa um elemento de acaso, que torne as decisões da banca imprevisíveis. Não devemos, porém, programar o computador com uma estratégia suicida — que faça a banca pedir sempre mais cartas, já tendo 20 pontos, por exemplo.

Criamos, assim, uma variável **R**, igual a 20, menos o total de pontos da banca. Um número qualquer entre 0 e 100 é escolhido e comparado com **R*R*R**. Se o número for menor, se a banca tiver menos de 16 pontos ou se o jogador conseguir uma mão de cinco, o computador pede mais uma carta. A maneira como calculamos **R** faz com que a chance da banca pedir cartas diminua à medida que seu total de pontos aumenta. Raramente ela pede mais uma carta já tendo 19 pontos, mas quase sempre o fará quando ainda tiver 16.

O RESULTADO FINAL

A linha 570 mostra o total de pontos do jogador. A linha 575 indica uma mão de cinco. As linhas 580 e 585 informam quais os tipos de mão que a banca está pagando.

Se o jogador tiver uma mão de cinco e a banca não, a linha 590, após uma pequena pausa, exibe na tela VOCÊ GANHOU. De modo semelhante, a linha 600 compara os totais do jogador e da banca; o resultado é dado pelas linhas 610 ou 630, dependendo do vencedor.

Quando o jogador ganha, a linha 640 acrescenta as fichas obtidas às que ele já possuía. Sempre que alguém consegue um natural, as cartas são embaralhadas. A linha 700 verifica o indicador de naturais e as linhas 710 a 740 embaralham as cartas.

A linha 660 entra em ação quando o jogador obtém um natural, verificando se a banca conseguiu a mesma coisa — neste caso, a banca ganha e o programa segue na linha 610. Caso contrário, a 670 informa a vitória do jogador.

A linha 690 avisa que as fichas do jogador acabaram.

Finalmente, as linhas 790 a 820 oferecem-lhe a opção de jogar outra vez.

O programa completo é tão extenso que invade a primeira página — MA — da tela do TK-2000. Assim, para concluir o programa, as partes anteriores devem ser carregadas na página dois. Digite **MP** antes de carregar o programa — incompleto ou depois de terminado. Nunca volte à **MA**.

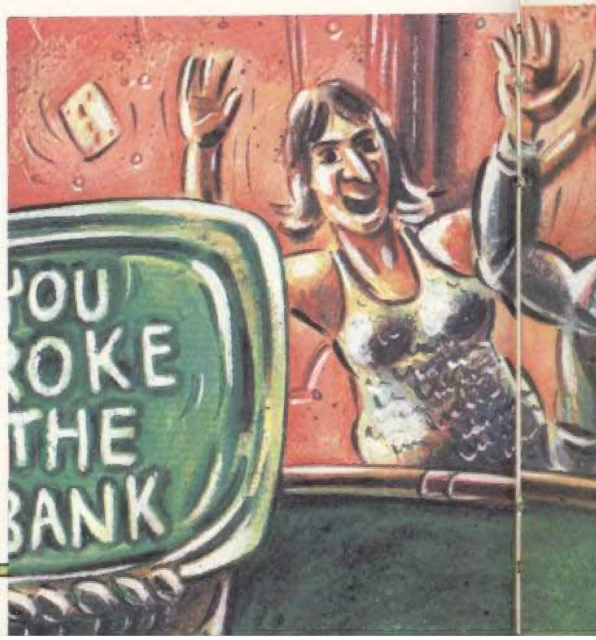
As alterações que os usuários do TK-2000 devem fazer para completar seu jogo são as mesmas do Apple, com exceção das seguintes linhas (não se esqueça de apagar o **GOTO 190** do final da linha 650):

```
540 DT = DL + 10 * (DA AND (DL
< 12)): IF DT = 21 AND NC = 2 T
HEN  GOSUB 5000: PRINT "A BANCA
 TEM UM NATURAL":PF = 1: GOSUB
6000: GOTO 610
550  IF NC = 5 THEN  GOSUB 5000
: PRINT "A BANCA TEM UMA MAO DE
 CINCO": GOSUB 6000: GOTO 610
570  GOSUB 5000: IF P5 = 1 THEN
  PRINT "VOCE TEM UMA MAO DE CI
NCO": GOSUB 6000: GOTO 580
575  GOSUB 5000: PRINT "VOCE TE
M ";PT: GOSUB 6000
580  GOSUB 5000: PRINT "A BANCA
 PAGA ";: IF DT < 21 THEN  PRIN
T "VALORES MAIORES OU IGUAIS A
";DT + 1: GOSUB 6000: GOTO 590
585  PRINT "NATURAIS E MAOS DE
CINCO APENAS": GOSUB 6000
610  GOSUB 5000: PRINT "A BANCA
 GANHOU": GOSUB 6000: GOTO 690
620  GOSUB 5000: PRINT "A BANCA
 ESTOUROU. VOCE GANHOU": GOSUB
6000: GOTO 640
630  GOSUB 5000: PRINT "VOCE GA
NHOU": GOSUB 6000
660  GOSUB 4000: GOSUB 5000: IF
 DL = 11 THEN  PRINT "MAS A BAN
CA TEM A MESMA COISA": GOSUB 60
00: GOTO 610
690  GOSUB 5000: IF MN < 1 THEN
  PRINT "VOCE PERDEU TODAS AS F
ICHAS": GOSUB 6000: GOTO 790
700  IF PF = 1 THEN  PRINT "EMB
ARALHANDO AS CARTAS": GOSUB 150
0: GOSUB 6000: GOTO 750
790  PRINT "QUER JOGAR NOVAMENT
E (S/N) ?"
3500  GOSUB 1000: GOSUB 2000: I
F NM = 1 THEN DA = 1
4000 NN = N:N = D1:CX = 6:CY =
110: GOSUB 3500
```

Você precisará, portanto, recorrer à listagem do Apple para fazer as demais modificações.

As diferenças entre o programa do TK-2000 e do Apple devem-se à ausência de uma página exclusiva para textos no primeiro computador. Assim, as mensagens têm que ser impressas na página gráfica.

As explicações sobre o funcionamento do programa são as mesmas, com uma exceção: na linha 3500 não ligamos a tela de alta resolução como no Apple, pois nunca saímos dela.



Antes de acrescentar as linhas seguintes, apague o **GOTO 190** do final da linha 650.

```
500 GOSUB 4000:GOTO 540
510 CX=CX+50:GOSUB 3500:NC=NC+1
520 FOR K=1 TO 1725:NEXT
530 IF DL>21 THEN 620
540 DT=DL+10*(DA AND(DL<12)):IF
 DT=21 AND NC=2 THEN CLS:PRINT"
 A BANCA TEM UM NATURAL":PF=1:G
OTO 610
550 IF NC=5 THEN PRINT" A BANCA
 TEM UMA MAO DE CINCO":GOTO 610
560 R=20-DT:IF RND(100)<R*R*R O
R DT<16 OR P5=1 THEN 510
570 CLS:IF P5=1 THEN PRINT" VOC
E TEM UMA MAO DE CINCO" ELSE PR
INT" VOCE TEM";PT
580 PRINT" A BANCA PAGA";:IF DT
<21 THEN PRINT DT+1 ELSE PRINT
" SOMENTE NATURAIS E MAOS DE
   CINCO"
590 FOR K=1 TO 500:NEXT:IF P5=1
 THEN 630
600 IF DT<PT THEN 630
610 PRINT" A BANCA GANHA":GOTO
690
620 CLS:PRINT" A BANCA ESTOUROU
. VOCE GANHA":GOTO 640
630 PRINT " VOCE GANHOU DA BANC
A"
640 MN=MN+2*BT:GOTO 700
660 GOSUB 4000:IF DL=11 THEN PR
INT:PRINT" MAS A BANCA OBTEVE O
 MESMO VALOR":GOTO 610
670 GOTO 630
690 PRINT:IF MN<1 THEN PRINT "
VOCE PERDEU TODAS AS FICHAS":GO
TO 790
700 IF PF=1 THEN PRINT:PRINT" A
 BANCA EMBARALHA AS CARTAS
APOS OBTER O NATURAL":GOSUB 150
0:GOTO 750
710 NF=N-1:IF NA>N THEN NF=N+51
720 FOR X=NF TO NA+1 STEP-1:Q=R
ND(X-NA)+NA-1:T=SQ(X):SQ(X)=SQ(
Q):SQ(Q)=T:NEXT:IF NA>N THEN 74
0
730 FOR X=0 TO 9:SQ(X+52)=SQ(X)
:NEXT:GOTO 750
740 FOR X=0 TO 9:SQ(X)=SQ(X+52)
:NEXT
790 PRINT:PRINT" QUER RECOMECAR
 (S/N)?"
800 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 800
810 IF A$="S" THEN CLS:GOTO 170
820 END
3500 SCREEN 1,1:GOSUB 1000:GOSU
B 2000:IF NM=1 THEN DA=1
3510 IF NM>10 THEN DL=DL+10 ELS
EDL=DL+NM
3520 RETURN
4000 NN=N:N=D1:CX=6:CY=108:GOSU
B 3500
4010 N=D2:CX=56:GOSUB 3500:N=NN
4020 FOR K=1 TO 2000:NEXT:NC=2:
RETURN
```

Quando a vez de jogar passa à banca, vêem-se na tela a mão do jogador e as duas cartas fechadas de seu adversário eletrônico. A banca, em primeiro lugar, abre suas duas cartas — a linha 500 chama a sub-rotina 4000, que abre a primeira carta chamando outra sub-rotina, na linha 3500. Esta liga a tela de alta resolução, desenha a carta e calcula os pontos. A segunda carta é aberta pela linha 4010, que chama a mesma sub-rotina após ter modificado a posição horizontal da carta. O número da carta é ajustado e a sub-rotina termina na linha 4020.

A linha 510 calcula a posição horizontal da nova carta e chama a sub-rotina da linha 3500. Assim, a banca recebe mais uma carta. A linha 520 provoca uma pausa antes que a 530 verifique se a banca tem mais de 21 pontos — a linha 620 informa o fato, se necessário. A linha 540 verifica se a banca conseguiu um natural; em caso afirmativo, coloca 1 no indicador de naturais e exibe uma mensagem na tela. Mãos de cinco são detectadas pela linha 550.

A parte mais interessante do programa está na linha 560, que faz o computador decidir se quer ou não mais cartas. Seria mais fácil programar a máquina de maneira que ela parasse de pedir cartas quando atingisse um determinado valor — 19, por exemplo. Mas seria igualmente fácil para o jogador derrotá-la, uma vez que tivesse conhecimento dessa estratégia de jogo. Por isso, é necessário introduzir no programa um elemento de acaso, que torne as decisões da banca imprevisíveis. Por outro lado, não podemos munir o programa de uma estratégia suicida, que faça, por exemplo, com que a banca freqüentemente peça mais cartas, já tendo obtido 20 pontos.

Criamos, assim, uma variável **R**, igual a 20, menos o valor da mão da banca. O programa escolhe ao acaso um número entre 1 e 100 e o compara com **R*R*R**. Se o número for menor, se a banca tiver menos de 16 pontos ou se o jogador conseguir uma mão de cinco, a banca pede mais uma carta. A maneira como calculamos **R** faz com que a chance da banca pedir cartas diminua à medida que seu total de pontos aumenta. Raramente ela pedirá mais cartas tendo já 19 pontos, mas quase sempre o fará quando tiver apenas 16.

O RESULTADO FINAL

A linha 570 mostra o total de pontos do jogador ou indica uma mão de cinco. A linha 580 informa quais os tipos de mão a banca vai pagar.

Se o jogador obteve uma mão de cinco e a banca não, a linha 590 exibe na tela a mensagem VOCÊ GANHOU. De modo semelhante, a linha 600 compara os totais do jogador e da banca; o resultado é dado pelas linha 610 ou 630, dependendo do vencedor.

Quando o jogador ganha, a linha 640 acrescenta as fichas obtidas às que ele possuía. Sempre que alguém obtém um natural, as cartas são embaralhadas pelas linhas 710 a 740. A linha 700 verifica o indicador de naturais.

A linha 660 entra em ação quando o jogador consegue um natural, verificando se a banca conseguiu mão igual. Se a banca vencer, a linha 610 anuncia. Caso contrário, a linha 630 informa a vitória do jogador.

A linha 690 avisa que o jogador perdeu todas as fichas.

Finalmente, as linhas 790 a 820 oferecem-lhe a opção de jogar novamente.

ROTINAS DE ORDENAÇÃO

Todo tipo de programa que manipula dados pode se beneficiar com o uso de uma rotina de ordenação. Neste artigo mostramos três dos métodos mais empregados para esse fim.

A ordenação é um dos aspectos fundamentais do processamento de informações. Os computadores são muito rápidos na manipulação de dados e a ordenação destes é um ponto importantíssimo para tornar a informação acessível à análise e correção.

Imagine como seria difícil encontrar as referências a um determinado assunto em um livro, sem o auxílio de um índice. Ou procurar um número numa lista telefônica sem que os nomes estivessem em ordem alfabética. Estamos, nos dois casos, diante de listas de informações, nas quais, ao contrário de uma lista de compras, por exemplo, os itens são arranjados ou ordenados numa ordem específica: a alfabética. Em outros casos — como na ordenação das notas de uma classe —, os itens estariam arranjados em ordem numérica.

Muitos tipos de programa fazem uso de rotinas de ordenação, inclusive os programas de jogos, que comparam placares. Essas rotinas, porém, são mais comumente encontradas em programas que lidam com grande quantidade de dados, como os de controle de arquivos, mala direta etc.

O QUE É ORDENAÇÃO

Ordenar é, simplesmente, colocar determinado conjunto de dados em uma ordem específica — algo como arrumar as cartas de um baralho.

A posição relativa de dois itens quaisquer em geral se baseia em uma superioridade numérica ou alfabética de um sobre outro.

Em ordenações numéricas, valores maiores vêm sucessivamente antes ou depois dos menores. Em ordenações alfabéticas, as letras são colocadas, sucessivamente, das mais próximas do início até as mais próximas do fim do alfabeto, ou vice-versa.

Pode-se também fazer ordenações em base alfanumérica, ou seja, levando-se em conta letras e números. Mas sempre haverá a prioridade de uns sobre outros — as letras quase sempre têm mais importância que os números.

- O QUE É ORDENAÇÃO?
- ORDENAÇÃO TIPO BOLHA
- AS VANTAGENS DA ROTINA DE ORDENAÇÃO DE BUSCA BINÁRIA
- AS ROTINAS DE SHELL E SHELL-METZNER
- COMO AUMENTAR A VELOCIDADE DE ORDENAÇÃO

Em ordenação numérica, em geral o número 1 tem prioridade, enquanto a letra A assume o primeiro lugar na ordenação alfabética. Mas para fins especiais isso pode ser invertido.

A ORDENAÇÃO TIPO BOLHA

A mais simples rotina para ordenação é a chamada ordenação tipo bolha. Você já viu um exemplo dela no artigo da página 292. Agora, examinaremos o processo em maior detalhe. Para acompanhar o que acontece, use cartas de baralho ou, então, recorte e numere alguns pedaços de papel.

Coloque esse grupo de cartas (ou pedaços de papel numerados) sobre uma mesa, com o três mais distante de você, na seguinte ordem:

última	[três]
	[sete]
	[dois]
	[seis]
primeira	[dez]

Em uma ordenação tipo bolha, o primeiro valor — aqui, o dez — é comparado com o seguinte do conjunto, havendo uma troca, se o valor deste for menor (o que ocorre no nosso caso). O dez é então comparado com o dois, com o sete e com o três, mudando sempre de lugar, porque é o maior valor a cada comparação dois a dois. Poderíamos dizer, usando uma imagem, que o dez "borbulhou" entre os outros valores, o que explica o nome desta rotina. Pelo mesmo raciocínio, valores baixos "borbulham" no sentido inverso.

Assim, depois da primeira passagem, o arranjo das cartas ficou assim:

última	[dez]
	[três]
	[sete]
	[dois]
primeira	[seis]

O processo reinicia com a nova primeira carta, o seis, que é, então, comparada e trocada com a anterior, visto

que tem valor maior. Mas, em seguida, é comparada com o sete que, por sua vez, é maior. As comparações continuam, então, com este valor, enquanto o seis fica na segunda posição. O sete é comparado com o três e depois com o dez, subindo apenas um posto, já que encontrou um valor maior.

A nova ordem das cartas, após a segunda passagem, é a seguinte:

última	[dez]
	[sete]
	[três]
	[seis]
primeira	[dois]

A primeira carta, o dois, é comparada com a anterior. O seis ganha e fica na mesma posição. Comparado com o três, o seis troca de lugar com ele, mas fica agora parado nesta posição, uma vez que é menor que o sete e o dez. A ordenação terminou, pois não há mais trocas a se fazer.

A ordem das cartas é, então:

última	[dez]
	[sete]
	[seis]
	[três]
primeira	[dois]

O computador faria uma última passagem por essa nova ordem das cartas, para se certificar de que não há mais trocas a fazer, e só pararia se realmente não houvesse.

Durante a execução da rotina um certo número de comparações e trocas foi feito. Essas duas operações estão incluídas em qualquer rotina de ordenação. A diferença entre tais rotinas consiste apenas no número de operações que são executadas e, conseqüentemente, no tempo gasto.

Vejamos agora outros exemplos, usando esta seqüência de números:

início							**fim**
67	35	72	19	47	38	11	96

O primeiro da lista é o 67. Ele é comparado e trocado com o 35, mas pára aí. O maior valor neste ponto é 72, que é comparado e trocado sucessivamente com 19, 47, 38, 11, até que encontra o 96; a rotina, então, recomeça a comparação desde o primeiro número.

A lista abaixo mostra todo o processo. Os itens submetidos a comparação estão entre parênteses.

```
( 67 )( 35 )  72    19    47    38    11    96
  35  ( 67 )( 72 )  19    47    38    11    96
  35    67  ( 72 )( 19 )  47    38    11    96
  35    67    19  ( 72 )( 47 )  38    11    96
  35    67    19    47  ( 72 )( 38 )  11    96
  35    67    19    47    38  ( 72 )( 11 )  96
  35    67    19    47    38    11  ( 72 )( 96 )
( 35 )( 67 )  19    47    38    11    72    96
  35  ( 67 )( 19 )  47    38    11    72    96
  35    19  ( 67 )( 47 )  38    11    72    96
  35    19    47  ( 67 )( 38 )  11    72    96
  35    19    47    38  ( 67 )( 11 )  72    96
  35    19    47    38    11  ( 67 )( 72 )  96
  35    19    47    38    11    67  ( 72 )( 96 )
( 35 )( 19 )  47    38    11    67    72    96
  19  ( 35 )( 47 )  38    11    67    72    96
  19    35  ( 47 )( 38 )  11    67    72    96
  19    35    38  ( 47 )( 11 )  67    72    96
  19    35    38    11  ( 47 )( 67 )  72    96
  19    35    38    11    47  ( 67 )( 72 )  96
  19    35    38    11    47    67  ( 72 )( 96 )
( 19 )( 35 )  38    11    47    67    72    96
  19  ( 35 )( 38 )  11    47    67    72    96
  19    35  ( 38 )( 11 )  47    67    72    96
  19    35    11  ( 38 )( 47 )  67    72    96
```

```
 19    35    11    38  ( 47 )( 67 )  72    96
 19    35    11    38    47  ( 67 )( 72 )  96
 19    35    11    38    47    67  ( 72 )( 96 )
( 19 )( 35 )  11    38    47    67    72    96
 19  ( 35 )( 11 )  38    47    67    72    96
 19    11  ( 35 )( 38 )  47    67    72    96
 19    11    35  ( 38 )( 47 )  67    72    96
 19    11    35    38  ( 47 )( 67 )  72    96
 19    11    35    38    47  ( 67 )( 72 )  96
 19    11    35    38    47    67  ( 72 )( 96 )
( 19 )( 11 )  35    38    47    67    72    96
 11  ( 19 )( 35 )  38    47    67    72    96
 11    19  ( 35 )( 38 )  47    67    72    96
 11    19    35  ( 38 )( 47 )  67    72    96
 11    19    35    38  ( 47 )( 67 )  72    96
 11    19    35    38    47  ( 67 )( 72 )  96
 11    19    35    38    47    67  ( 72 )( 96 )
( 11 )( 19 )  35    38    47    67    72    96
 11  ( 19 )( 35 )  38    47    67    72    96
 11    19  ( 35 )( 38 )  47    67    72    96
 11    19    35  ( 38 )( 47 )  67    72    96
 11    19    35    38  ( 47 )( 67 )  72    96
 11    19    35    38    47  ( 67 )( 72 )  96
 11    19    35    38    47    67  ( 72 )( 96 )
```

Vejamos agora como se comporta a rotina de ordenação tipo bolha na forma de programa. Ela está colocada como a sub-rotina 1000.

Grave o programa para que possa adicionar outras rotinas mais tarde.

```
 10 POKE 23658,8: LET T=0:
 INPUT "NUMERO DE ITENS ";AA:
 IF AA<2 THEN  GOTO 10
 15 DIM A(AA)
 20 PRINT '''"TABELA DESORDENA
 DA": PRINT
 30 FOR Z=1 TO AA
 40 LET A(Z)=INT (RND*100)+1
 50 PRINT TAB T;A(Z);: LET T=T
 +4: IF T>30 THEN  LET T=0
 60 NEXT Z
 70 PRINT : PRINT : PRINT "PRE
 SSIONE 'O' PARA ORDENAR"
 80 LET K$=INKEY$: IF K$<>"O"
 THEN  GOTO 80
 90 GOSUB 1000
100 PRINT : PRINT "TABELA ORDE
NADA": PRINT
110 LET T=0: FOR Z=1 TO AA
120 PRINT TAB T;A(Z);: LET T=T
+4: IF T>30 THEN  LET T=0
130 NEXT Z
140 GOTO 10
999 REM ORDENACAO BOLHA(BUBBLE
SORT)
1000 FOR Z=1 TO AA-1
1010 LET ZZ=0
1020 FOR Y=1 TO AA-Z
1030 IF A(Y+1)<A(Y) THEN  LET X
=A(Y): LET A(Y)=A(Y+1): LET A(Y
+1)=X: LET ZZ=1
1040 NEXT Y
1050 IF ZZ=0 THEN  RETURN
1060 NEXT Z: RETURN
```

```
10 PRINT:PRINT:INPUT"NUMERO DE
ITENS";AA:IF AA<2 THEN 10
15 DIM A(AA)
20 PRINT:PRINT"TABELA DESORDENA
DA":PRINT
30 FOR Z=1 TO AA
40 A(Z)=INT(RND(1)*100)+1
50 PRINT A(Z),
60 NEXT Z
70 PRINT:PRINT:PRINT"PRESSIONE
'O' PARA ORDENAR"
80 GET K$:IF K$<>"O" THEN 80
90 GOSUB 1000
100 PRINT:PRINT:PRINT"TABELA OR
DENADA"
110 PRINT:FOR Z=1 TO AA
120 PRINT A(Z),
130 NEXT Z
140 RUN
999 REM ORDENACAO BOLHA(BUBBLE
SORT)
1000 FOR Z=1 TO AA-1
1010 ZZ=0
1020 FOR Y=1 TO AA-Z
1030 IF A(Y+1)<A(Y) THEN X=A(Y)
:A(Y)=A(Y+1):A(Y+1)=X:ZZ=1
1040 NEXT Y
1050 IF ZZ=0 THEN RETURN
1060 NEXT Z:RETURN
```

O programa acima roda apenas no Apple e no TK-2000. Faça as seguintes alterações para adaptar o programa à sua máquina:

```
40 A(Z)=RND(100)
50 PRINT A(Z);
80 K$=INKEY$:IF K$<>"O" THEN 80
120 PRINT A(Z);
```

```
5 R=RND(-TIME)
80 K$=INKEY$:IF K$<>"O" THEN 80
1030 IF A(Y+1)<A(Y) THEN SWAP A
(Y),A(Y+1):ZZ=1
```

Execute o programa. Inicialmente, ele pedirá a quantidade de itens que você quer ordenar, criando um conjunto de números pseudo-randômicos baseado na sua resposta. Estes são mostrados na tela sob o título "TABELA DESORDENADA". Uma mensagem pede que você tecle '0' para iniciar a ordenação.

A rotina começa a funcionar, concluindo a ordenação em cerca de um segundo, se os itens forem poucos. Em seguida, o conjunto de números é exibido na tela. No artigo da página 292 você encontrará mais detalhes sobre esta rotina e verá também o algoritmo em forma de diagrama.

Para que você possa comparar as diversas rotinas, é interessante que cronometre o tempo gasto em cada uma delas. Incorpore a rotina seguinte ao programa, a fim de que o computador faça isso para você. Os usuários do Apple II ou compatível precisarão utilizar o seu relógio...

```
90 TIMER=0:GOSUB 1000:PRINT:PRI
NT" TEMPO: ";TIMER/50;"SEGUNDOS
"
```

```
90 POKE 16920,0:POKE 16919,0:GO
SUB 1000:PRINT"Tempo:";PEEK(169
20);"m";PEEK(16919);"s"
```

```
90 POKE 23672,0: POKE 23673,0
: GOSUB 1000: PRINT : PRINT (
PEEK 23672+256*PEEK 23673)/50
;" SEGUNDOS"
```

```
90 TIME=0:GOSUB1000:PRINT:PRINT
"Tempo: ";TIME/60;" segundos"
```

Para iniciar a cronometragem e a ordenação, pressione '0' quando a mensagem aparecer. O tempo gasto pela rotina para ordenar os dados é exibido e o programa recomeça automaticamente.

Provavelmente você gastaria muito mais tempo do que a máquina para executar a mesma tarefa. Porém, pelos padrões computacionais, o tempo gasto pela ordenação tipo bolha é extremamente longo. Por essa razão, não se costuma usá-la em sua forma original em programas de manipulação de dados, a não ser para listas muito pequenas.

Mas veja que, surpreendentemente, ela se mostra mais veloz que todas as outras quando se trata de reordenar uma lista à qual se adicionou um item. Em um programa que armazena endereços comerciais, por exemplo, podemos ordenar as novas entradas separadamente e depois incorporá-las à lista principal. Outra alternativa é reordenar a lista toda a cada nova entrada.

Nessas circunstâncias, a rotina tipo bolha não faz nada mais que comparar o novo dado com os outros, até que sua posição correta seja encontrada. A reordenação de uma lista equivale, assim, à comparação de dois itens. E isso é feito rapidamente.

A ROTINA DE SHELL

As duas versões mais utilizadas da rotina tipo bolha são as rotinas de Shell e de Shell-Metzner. Ambas devem ser consideradas quando o número de itens a serem ordenados ultrapassa a casa dos dez.

A rotina de Shell emprega uma técnica de busca binária que funciona dividindo sucessivamente a lista original ao meio, até que a posição do item em questão seja encontrada. A cada divi-

são, um novo par de valores é comparado. A rotina termina quando faz uma passagem completa sem trocas.

Uma rotina de Shell tem, tipicamente, a forma mostrada aqui. Adicione-a ao seu programa e mude o número do **GOSUB** da linha 90 para testá-la.

```
1999 REM ORDENACAO SHELL
2000 LET Z=AA
2010 IF Z<=1 THEN  RETURN
2020 LET Z=INT (Z/2): LET Y=AA-
Z
2030 LET ZZ=0
2040 FOR X=1 TO Y
2050 LET XX=X+Z
2060 IF A(X)>A(XX) THEN  LET YY
=A(X): LET A(X)=A(XX): LET A(XX
)=YY: LET ZZ=1
2070 NEXT X
2080 IF ZZ>0 THEN  GOTO 2030
2090 GOTO 2010
```

```
1999 REM ORDENACAO SHELL
2000 Z=AA
2010 IF Z<=1 THEN RETURN
2020 Z=INT(Z/2):Y=AA-Z
2030 ZZ=0
2040 FOR X=1 TO Y
2050 XX=X+Z
2060 IF A(X)>A(XX) THEN YY=A(X)
:A(X)=A(XX):A(XX)=YY:ZZ=1
2070 NEXT X
2080 IF ZZ>0 THEN 2030
2090 GOTO 2010
```

MICRO DICAS

PARA UMA ORDENAÇÃO MAIS RÁPIDA

A velocidade de qualquer rotina de ordenação aumenta muito quando a informação é oferecida já semi-ordenada. Isso é importante sobretudo no caso da ordenação tipo bolha.

Se os itens vão ser ordenados cronologicamente, por exemplo, devemos armazenar as datas na forma ANO/MÊS/DIA, em vez da mais usual DIA/MÊS/ANO. Dessa maneira, aplica-se a ordenação inicialmente ao grupo completo e, depois, aos subgrupos.

O processo seria muito mais demorado se ordenássemos inicialmente os dias, para depois reordenar alguns dados em função dos meses, repetindo finalmente todo o trabalho em função dos anos.

Uma sub-rotina adicional poderia ser usada para inverter a seqüência usual de entrada da data para uso da rotina de ordenação.

Se o seu micro é da linha MSX, mude a linha 2060 para:

```
2060 IF A(X)>A(XX) THEN SWAP A(
X),A(XX):ZZ=1
```

Para melhor compreender o funcionamento da rotina, vamos examinar em detalhe um exemplo. Suponhamos que devemos ordenar uma lista de números apresentada nesta ordem:

67 35 72 19 47 38 11 96

A rotina divide a lista em duas outras e compara o primeiro valor de cada uma delas. Se o primeiro valor da primeira lista é maior, ocorre uma troca. Neste exemplo, os valores 67 e 47 são comparados e trocados.

47 35 72 19 : 67 38 11 96

Após a troca, o par de valores seguinte é comparado — 35 e 38. Nesse caso, não há troca. A rotina compara, então, 72 e 11, trocando-os e, por fim, 19 e 96, obviamente sem troca. A lista fica assim:

47 35 11 19 : 67 38 72 96

Neste ponto, cada metade é novamente dividida:

47 35 : 11 19 : 67 38 : 72 96

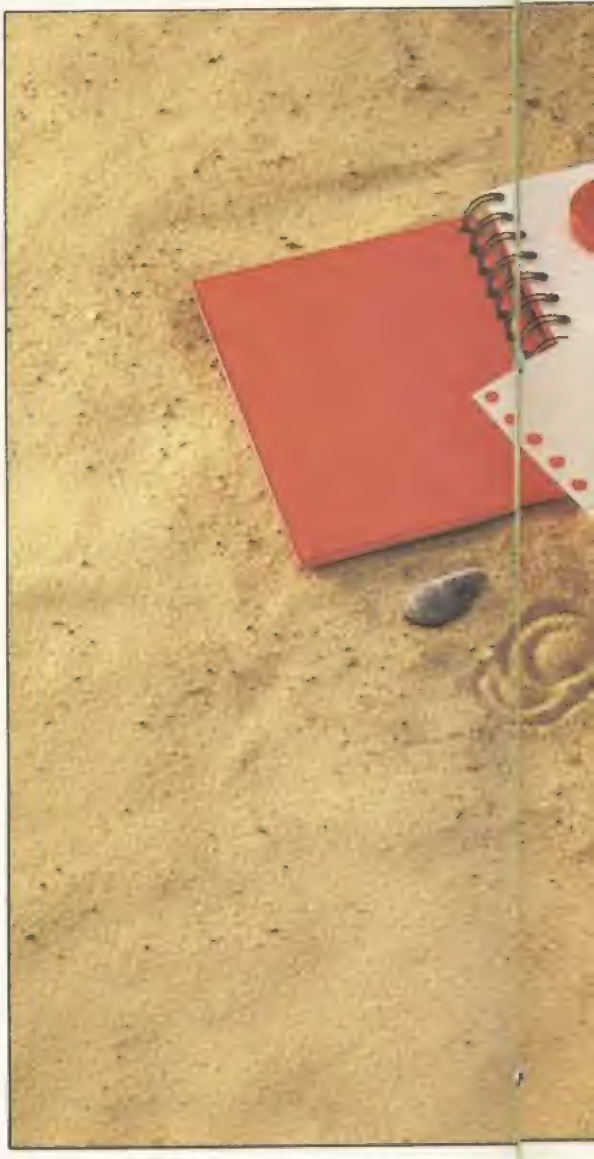
A rotina compara e troca o primeiro valor, 47, com o terceiro, 11. Depois, compara e troca o segundo, 35, com o quarto, 19. O 35 assume agora a quarta posição. O valor no terceiro posto, 47, que já foi trocado uma vez, é comparado com 67. Desta vez não há troca. O quarto valor, 35, é comparado com 38 e permanece onde está. O quinto valor, 67, é comparado com o sétimo, 72, e fica na mesma posição. Por fim, 38 e 96 são comparados e não trocam de lugar.

Veja abaixo a seqüência. Como no exemplo anterior, os valores comparados estão entre parênteses. Verifique se, na linha seguinte, alguma troca foi efetuada:

```
(47) 35 : (11) 19 : 67 38 : 72 96
 11 (35) : 47 (19) : 67 38 : 72 96
 11 19 : (47) 35 : (67) 38 : 72 96
 11 19 : 47 (35) : 67 (38) : 72 96
 11 19 : 47 35 : (67) 38 : (72) 96
 11 19 : 47 35 : 67 (38) : 72 (96)
 11 19 : 47 35 : 67 38 : 72 96
```

A lista é, então, dividida da seguinte maneira:

11 : 19 : 47 : 35 : 67 : 38 : 72 : 96



Os valores são novamente comparados dois a dois. Como você pode observar, a rotina compara somente valores adjacentes: o primeiro com o segundo, o segundo com o terceiro, o terceiro com o quarto e assim por diante, até o fim da lista:

```
(11) : (19) : 47 : 35 : 67 : 38 : 72 :96
 11  : (19) :(47): 35 : 67 : 38 : 72 :96
 11  :  19  :(47):(35): 67 : 38 : 72 :96
 11  :  19  : 35 :(47):(67): 38 : 72 :96
 11  :  19  : 35 : 47 :(67):(38): 72 :96
 11  :  19  : 35 : 47 : 38 :(67):(72):96
```

A rotina faz o mesmo até os últimos valores, que já estão em ordem, e volta ao início:

```
(11) : (19) : 35 : 47 : 38 : 67 : 72 :96
 11  : (19) :(35): 47 : 38 : 67 : 72 :96
 11  :  19  :(35):(47): 38 : 67 : 72 :96
```

11 : 19 : 35 :(47):(38): 67 : 72 :96
11 : 19 : 35 : 38 :(47):(67): 72 :96

Embora a lista já esteja ordenada, o computador executa mais uma passagem completa. Cada novo valor pode agora ser comparado com essa lista como o faz a rotina tipo bolha — isto é, estabelecendo comparações aos pares.

A ROTINA DE SHELL-METZNER

A rotina de ordenação de Shell-Metzner, derivada da rotina de Shell, é ainda mais rápida e utiliza o mesmo princípio de busca binária.

```
2999 REM SHELL-MELTZNER SORT
3000 LET Z=AA
3010 LET Z=INT (Z/2)
3020 IF Z=0 THEN  RETURN
3030 LET Y=AA-Z: LET ZZ=1
3040 LET X=ZZ
3050 LET XX=X+Z
3060 IF A(X)<=A(XX) THEN  GOTO
 3090
3070 LET W=A(X): LET A(X)=A(XX)
: LET A(XX)=W: LET X=X-Z
3080 IF X>=1 THEN  GOTO 3050
3090 LET ZZ=ZZ+1
3100 IF ZZ<=Y THEN  GOTO 3040
3110 GOTO 3010
```

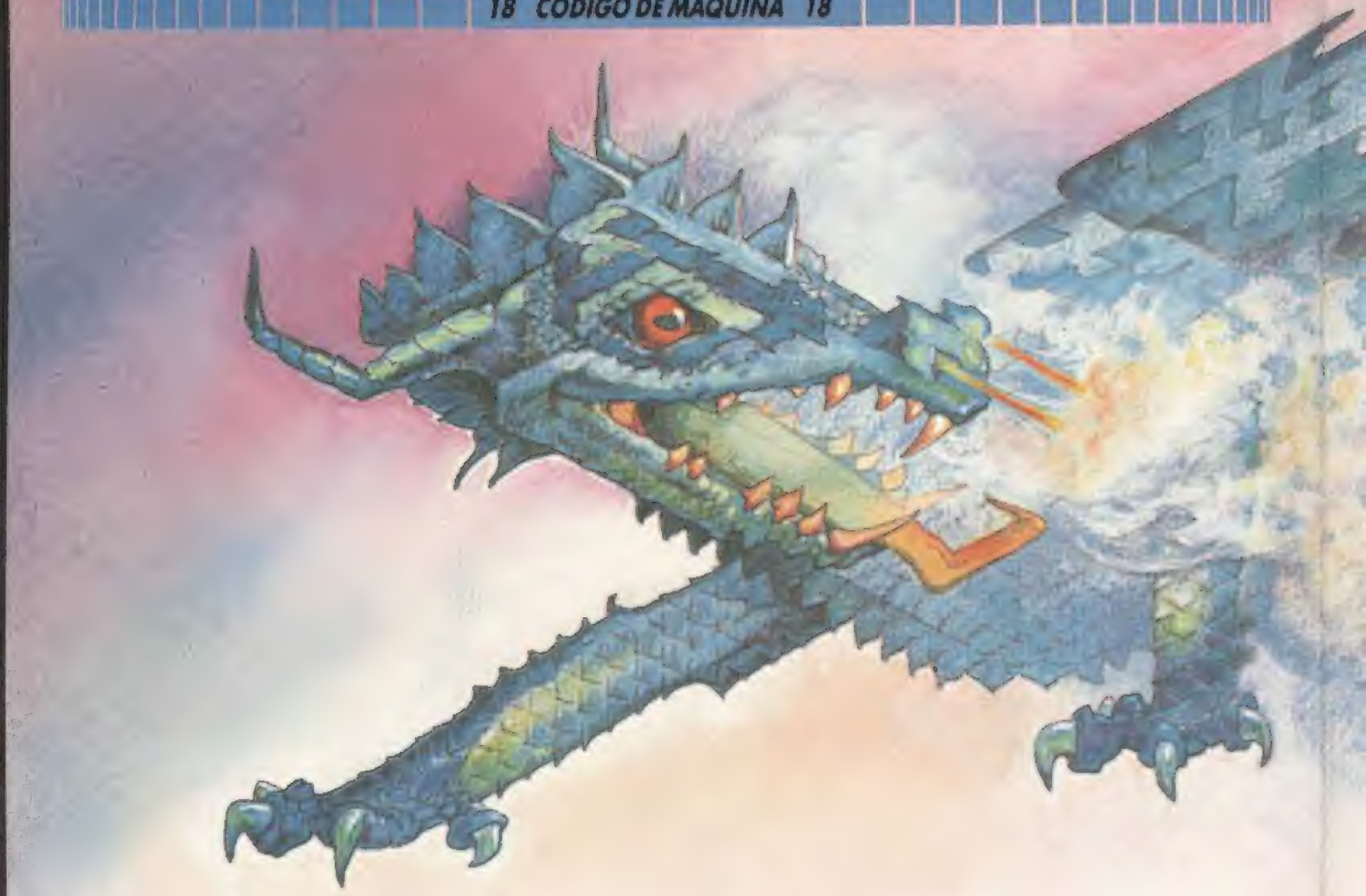
```
2999 REM ORDENACAO 'SHELL-METZN
ER'
3000 LET Z=AA
3010 Z=INT(Z/2)
3020 IF Z=0 THEN RETURN
3030 Y=AA-Z:ZZ=1
3040 X=ZZ
3050 XX=X+Z
3060 IF A(X)<=A(XX) THEN 3090
3070 W=A(X):A(X)=A(XX):A(XX)=W:
X=X-Z
3080 IF X>=1 THEN 3050
3090 ZZ=ZZ+1
3100 IF ZZ<=Y THEN 3040
3110 GOTO 3010
```

Se você trabalha com um MSX, beneficie-se de seu poderoso BASIC, trocando a linha 3070 para:

```
3070 SWAP A(X),A(XX):X=X-Z
```

Adicione esta rotina ao seu programa de demonstração e para experimentá-lo será preciso modificar a referência de linha da linha 90.

Como você vai notar, ela é muito mais rápida que as outras rotinas, o que fica evidente especialmente se a compararmos com a rotina tipo bolha para mais de cinqüenta números.



DRAGÃO ANIMADO

Aventura que se preze, com reis, rainhas e castelos, não dispensa a participação de um temível dragão.

Não é difícil incluir esse personagem em seus jogos. Para criá-lo e dar-lhe animação — fazendo com que cuspa fogo —, você pode utilizar as técnicas já vistas nos programas do sapo e do tanque, apresentados no artigo publicado à página 341.

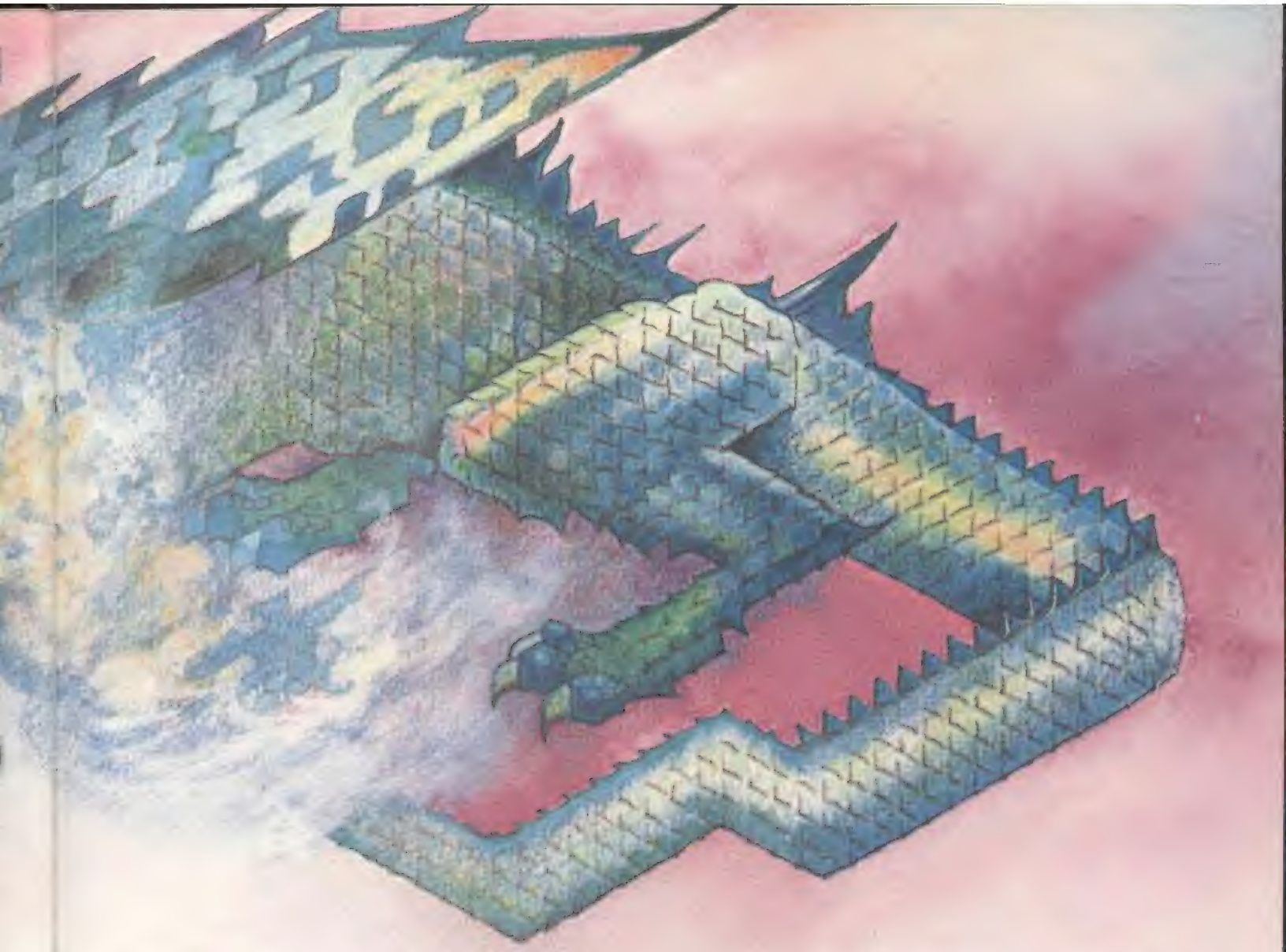
Para montar o "quadriculado" na memória do computador, usaremos o programa dado no artigo anterior. O Apple, o TK-2000 e também o MSX não precisam desse tipo de programa. A listagem para o Spectrum e o TRS-Color encontra-se naquele artigo.

Depois de carregar — ou digitar — o programa criador de quadriculado, use **RUN** para fazê-lo funcionar. Quando o programa acabar de rodar, digite **NEW** e **<ENTER>**. (Nada disso se aplica ao MSX, ao Apple e ao TK-2000.) Em seguida, digite o programa adequado para seu computador.

Os usuários do Apple, do TK-2000 e do MSX precisarão carregar — ou digitar — o programa do tanque, pois as linhas apresentadas neste artigo são modificações a serem feitas naquele programa, e não funcionarão sozinhas.

Em todos os computadores, exceto no MSX, use as teclas **P**, **L**, **X** e **Z** para mover o dragão. No MSX devem ser utilizadas as teclas do cursor. A barra de espaço faz o dragão cuspir fogo.

```
10 FOR n=USR "a" TO USR "t"+7
: READ a: POKE n,a: NEXT n
20 LET print=32400: LET b=
32402: IF PEEK 23733=255 THEN
LET print=65200: LET b=65202
90 BORDER 7: PAPER 7: INK 4:
CLS : PRINT AT 8,15;: RAND
USR print
100 LET y=8: LET x=15: LET y1=
8: LET x1=15: LET z=1
```

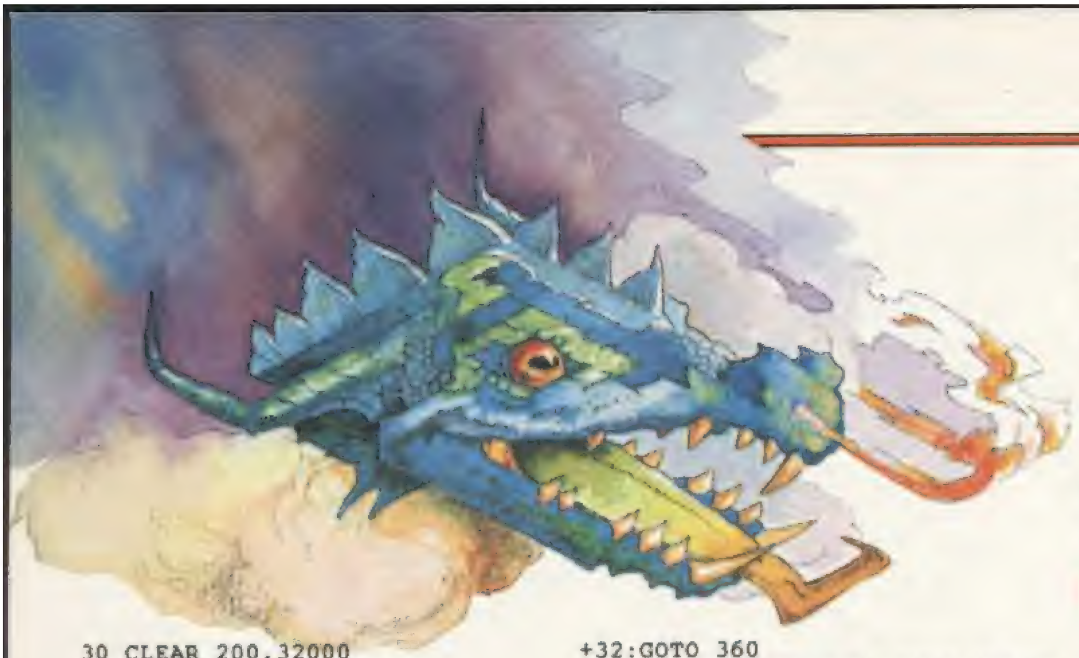


Vimos, em artigo anterior, como desenhar e dar movimento a um tanque de guerra e a um sapo. Com as técnicas e programas já apresentados, desenhe também um dragão que cospe fogo.

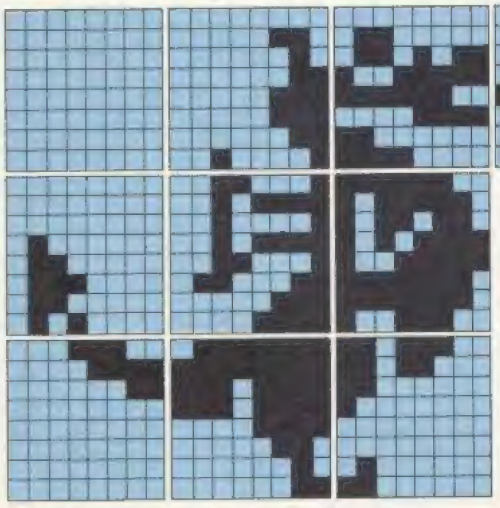
- COMO COLOCAR UMA NOVA FIGURA NO QUADRICULADO
- MOVIMENTOS NA TELA
- FAÇA O DRAGÃO CUSPIR FOGO

```
110 LET a$=INKEY$: IF a$=""
THEN  GOTO 110
115 IF a$=" " THEN  GOTO 200
120 IF a$="z" AND x>1 THEN
LET x1=x-1: LET z=1
130 IF a$="x" AND x<28 THEN
LET x1=x+1: LET z=2
140 IF a$="p" AND y>0 THEN
LET y1=y-1
150 IF a$="l" AND y<19 THEN
LET y1=y+1
160 PRINT AT y,x;: POKE b,0:
RAND USR print
170 LET x=x1: LET y=y1
180 PRINT AT y,x;: POKE b,z:
RAND USR print
190 GOTO 110
200 IF z=2 THEN  GOTO 300
220 PRINT  INK 6;AT y,x-1;CHR$
162
230 PAUSE 1
240 PRINT AT y,x-1;" "
260 GOTO 110
300 PRINT  INK 6;AT y,x+3;CHR$
163
310 PAUSE 1
320 PRINT AT y,x+3;" "
330 GOTO 110
1000 DATA 6,214,249,63,240,0,3,
15,96,64,192,224,224,192,196,20
4,0,0,0,0,0,0,0,0
1010 DATA 63,125,107,245,249,12
7,127,51,244,196,194,255,128,19
2,224,240,0,0,0,2,6,14,6,10
1020 DATA 55,23,7,3,1,0,4,7,254
,255,239,239,224,192,128,128,56
,240,192,0,0,0,0,0
1030 DATA 0,0,0,0,0,0,0,0,6,2,3
,7,7,3,35,51,96,107,159,252,15,
0,192,240
1040 DATA 0,0,0,64,96,112,96,80
,47,33,65,255,1,3,7,15,252,190,
182,175,159,254,254,204
1050 DATA 28,15,3,0,0,0,0,0,127
,255,247,247,7,3,1,1,236,232,22
4,192,128,0,32,224
1060 DATA 8,68,50,139,50,68,8,0
,16,34,76,145,76,34,16,0
5000 FOR i=1 TO 5
5010 SAVE "MC0301"
5020 NEXT i
```

```
20 PCLEAR 5
```



```
30 CLEAR 200,32000
40 FOR I=32300 TO 32587
50 READ N
60 POKE I,N
70 NEXT
80 DIM A(2),B(2),C(2)
90 PMODE 4,1
100 PCLS
110 FOR I=1536 TO 1760 STEP 32
120 READ N
130 POKE I,N
140 NEXT
150 GET (0,0)-(7,7),A
160 FOR I=1536 TO 1760 STEP 32
170 READ N
180 POKE I,N
190 NEXT
200 GET (0,0)-(7,7),B
210 PCLS
220 SCREEN 1,0
230 T=1
240 DP=3500
250 POKE 32700,INT(DP/256)
260 POKE 32701,DP-256*INT(DP/25
6)
270 POKE 32250,T+DT*2
280 EXEC 32000
290 LP=DP
300 IF PEEK(338)=251 THEN DP=DP
-32:GOTO 360
310 IF PEEK(342)=253 THEN DP=DP
+32:GOTO 360
320 IF PEEK(340)=247 THEN DP=DP
-1:T=T+1:DT=1:GOTO 360
330 IF PEEK(338)=247 THEN DP=DP
+1:T=T+1:DT=0:GOTO 360
340 IF INKEY$=" " AND T=2 GOSUB
 410
350 GOTO 300
360 IF DP<1536 OR DP>6941 THEN
DP=LP
370 IF T>2 THEN T=1
380 POKE 32250,0
390 EXEC 32000
400 GOTO 250
410 IF DT=1 GOTO 470
420 YP=INT((DP-1536)/32)
430 XP=8*(DP-1533-YP*32)
440 IF XP>255 THEN 530
450 PUT (XP,YP)-(XP+7,YP+7),A
460 GOTO 510
470 YP=INT((DP-1536)/32)
480 XP=8*(DP-1537-YP*32)
490 IF XP<0 THEN 530
500 PUT (XP,YP)-(XP+7,YP+7),B
510 FOR I=1 TO 200:NEXT
520 PUT(XP,YP)-(XP+7,YP+7),C
530 RETURN
540 DATA 0,0,0,0,0,0,0,0,6,2,3,
7,7,3,35,51,96,107,159,252,15,0
,192,240
550 DATA 0,0,0,0, 64, 96, 112, 9
6,47,33,65,255,3,7,
15, 31, 248,188 182,
175,159,255,254,254
560 DATA
80,28,15,7,1,0,0,0,
63,255,255,255,255,
15,2,3,252,248,252,
252, 248, 48,82,222
570 DATA 0,0,0,0,0,0,0,0,0,6,2,
3,7,7,3,33,0,96,107,159,252,15,
192,240
580 DATA 0,0,0,64,96,112,96,80,
49,47,33,47,49,99,7,15,252,190,
182,175,159,254,254,156
590 DATA 28,15,3,0,0,0,0,0,127,
255,239,239,15,6,2,3,220,216,19
2,128,0,0,64,192
600 DATA 6,214,249,63,240,0,3,1
5,96,64,192,224,224,192,196,204
,0,0,0,0,0,0,0,0
610 DATA 31,61,109,117,249,255,
127,127,244,196,194,255,192,224
,240,248,0,0,0,0,2,6,14,6
620 DATA 63,31,63,63,31,12,74,1
23,252,255,255.255,255.240,64,1
92,10,56,240,224,128,0,0,0
630 DATA 0,6,214,249,63,240,3,1
5,0,96,64,192,224,224,192,132,0
,0,0,0,0,0,0,0
640 DATA 63,125,109,245,249,127
,127,57,140,244,132,244,140,198
,224,240,0,0,0,2,6,14,6,10
650 DATA 59,27,3,1,0,0,2,3,254,
255,247,247,240,96,64,192,56,24
0,192,0,0,0,0,0
660 DATA 0,16,34,76,145,76,34,1
6
670 DATA 0,8,68,50,137,50,68,8
```



O dragão do TRS-Color é formado por blocos gráficos de oito por oito pontos.

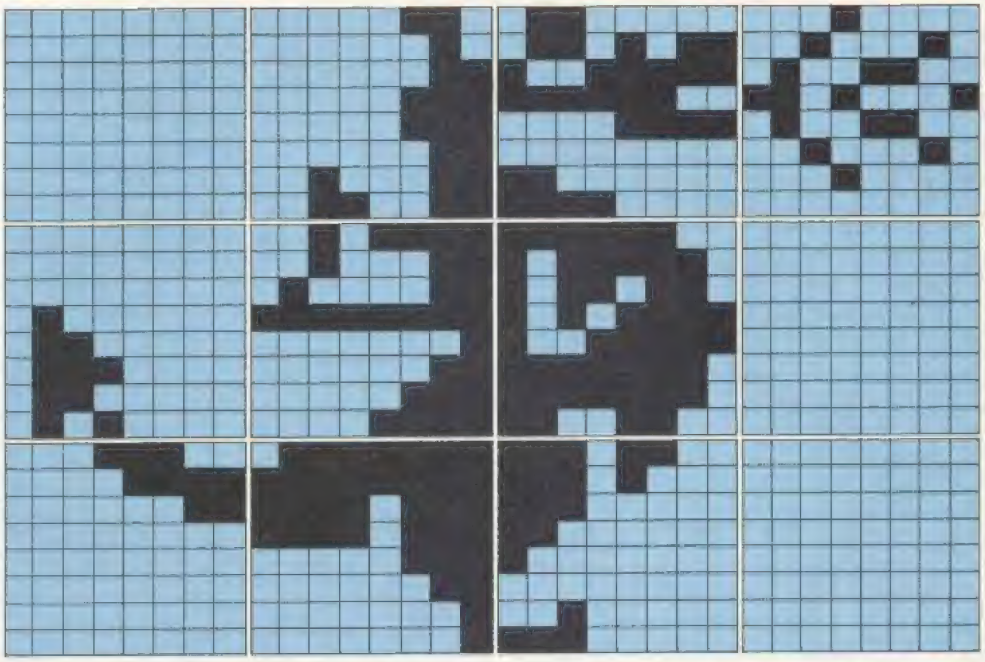
As linhas a seguir são modificações que devem ser feitas no programa que move o tanque de guerra.

```
5 CLEAR 1000:COLOR 12,15,12
20 FOR I=1 TO 32*4
25 READ A:A$=A$+CHR$(A)
30 NEXT I
35 FOR I=1 TO 32*4
40 READ B:B$=B$+CHR$(B)
45 NEXT I
50 SPRITE$(0)=LEFT$(A$,32)
55 FOR I=1 TO 3
60 SPRITE$(I)=MID$(A$,I*32+1,32
)
65 NEXT I
70 SPRITE$(4)=LEFT$(B$,32)
75 FOR I=5 TO 7
80 SPRITE$(I)=MID$(B$,(I-4)*32+
1,32)
85 NEXT I
110 X=100:Y=90:F=1
111 PUTSPRITE0,(X+16,Y-8),12,0
112 PUTSPRITE1,(X+16,Y+8),12,1
113 PUTSPRITE2,(X,Y+8),12,2
500 PUTSPRITE0,(X+16,Y-8),12,0
510 PUTSPRITE1,(X+16,Y+8),12,1
515 PUTSPRITE2,(X,Y+8),12,2
516 PUTSPRITE3,(X+32,209),12,3
600 PUTSPRITE0,(X+16,Y-8),12,4
610 PUTSPRITE1,(X+16,Y+8),12,5
615 PUTSPRITE2,(X+32,Y+8),12,6
616 PUTSPRITE3,(X,209),12,7
700 PUTSPRITE0,(X+16,Y-8),12,0
710 PUTSPRITE1,(X+16,Y+8),12,1
715 PUTSPRITE2,(X,Y+8),12,2
716 PUTSPRITE3,(X+32,Y),6,3
800 PUTSPRITE0,(X+16,Y-8),12,4
810 PUTSPRITE1,(X+16,Y+8),12,5
815 PUTSPRITE2,(X+32,Y+8),12,6
816 PUTSPRITE3,(X,Y),6,7
5000 DATA 0,0,0,0,0,0,0,0,6,2,3
,7,7,3,35,51,0,0,0,0,0,0,0,0,96
,107,159,252,15,0,192,240
5010 DATA 47,35,67,255,1,3,7,15
,127,255,247,247,7,3,1,1,252,19
0,182,175,159,254,254,204,232,2
24,224,192,128,0,32,224
5020 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,64,96,112,96,8
0,28,15,3,0,0,0,0,0
5030 DATA 16,34,76,209,76,34,16
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0
```

```
5040 DATA 0,0,0,0,0,0,0,0,6,214
,249,63,240,0,1,7,0,0,0,0,0,0,0
,0,96,64,192,224,224,192,196,20
4
5050 DATA 63,125,109,245,249,12
7,127,51,23,7,3,1,0,0,4,7,244,1
96,194,255,128,192,224,240,254,
255,239,239,224,128,128,192
5060 DATA 0,0,0,2,6,14,6,10,56,
240,192,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0
5070 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,8,68,50,139,50,68,8,
0,0,0,0,0,0,0,0,0,0
```

Carregue o programa que move um tanque na tela usando **DRAW**; faça nele as seguintes modificações:

```
30  FOR I = E TO E + 41 + 18 *
32
110  IF K$ = "Z" AND X > 16 THE
N X = X - 3:F = 0: GOTO 140
170  DRAW 1 AT LX,LY + 8
180  DRAW 2 AT LX,LY + 16
190  DRAW 3 AT LX + 8,LY
200  DRAW 4 AT LX + 8,LY + 8
205  DRAW 5 AT LX + 8,LY + 16
210  DRAW 6 AT LX + 16,LY
220  DRAW 7 AT LX + 16,LY + 8
225  DRAW 8 AT LX + 16,LY + 16
260  DRAW 11 AT LX,LY
270  DRAW 12 AT LX,LY + 8
275  DRAW 13 AT LX,LY + 16
280  DRAW 14 AT LX + 8,LY
290  DRAW 15 AT LX + 8,LY + 8
295  DRAW 16 AT LX + 8,LY + 16
300  DRAW 17 AT LX + 16,LY + 8
310  DRAW 18 AT LX + 16,LY + 16
340  DRAW 1 AT X,Y + 8
350  DRAW 2 AT X,Y + 16
360  DRAW 3 AT X + 8,Y
370  DRAW 4 AT X + 8,Y + 8
375  DRAW 5 AT X + 8,Y + 16
380  DRAW 6 AT X + 16,Y
390  DRAW 7 AT X + 16,Y + 8
395  DRAW 8 AT X + 16,Y + 16
420  DRAW 11 AT X,Y
430  DRAW 12 AT X,Y + 8
435  DRAW 13 AT X,Y + 16
440  DRAW 14 AT X + 8,Y
450  DRAW 15 AT X + 8,Y + 8
455  DRAW 16 AT X + 8,Y + 16
460  DRAW 17 AT X + 16,Y + 8
470  DRAW 18 AT X + 16,Y + 16
510  DRAW 9 AT X + 30,Y
540  DRAW 9 AT X + 30,Y
570  DRAW 10 AT X - 12,Y
600  DRAW 10 AT X - 12,Y
2000  DATA  20 ,0 ,42 ,0 ,74 ,0
 ,106 ,0 ,138 ,0 ,170 ,0 ,202 ,
0 ,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1
,106 ,1 ,138 ,1 ,170 ,1 ,202 ,1
 ,234 ,1 ,10 ,2 ,42 ,2 ,74 ,2 ,
106 ,2 ,138 ,2
2010  DATA  0 ,72 ,73 ,73 ,218
 ,219 ,219 ,74 ,73 ,73 ,218 ,21
9 ,27 ,87 ,109 ,73 ,209 ,219 ,5
9 ,191 ,41 ,77 ,73 ,218 ,219 ,3
1 ,23 ,0 ,0 ,0 ,0 ,0
```



O Spectrum, o MSX, o Apple e o TK-2000 utilizam este dragão como modelo.

```
2020  DATA  0 ,72 ,41 ,109 ,20
9 ,63 ,255 ,155 ,73 ,73 ,173 ,2
19 ,219 ,155 ,73 ,73 ,137 ,219
,219 ,155 ,73 ,73 ,137 ,219 ,21
9 ,155 ,0 ,0 ,0 ,0 ,0 ,0
2030  DATA  0 ,72 ,73 ,109 ,21
8 ,223 ,219 ,74 ,73 ,41 ,213 ,6
3 ,223 ,155 ,73 ,9 ,45 ,213 ,25
5 ,219 ,83 ,105 ,9 ,173 ,59 ,22
3 ,255 ,2 ,0 ,0 ,0 ,0
2040  DATA  0 ,72 ,13 ,45 ,173
 ,59 ,223 ,251 ,10 ,77 ,9 ,173
,59 ,63 ,63 ,63 ,78 ,73 ,9 ,213
 ,255 ,219 ,83 ,73 ,41 ,173 ,59
 ,63 ,223 ,19 ,0 ,0
2050  DATA  0 ,8 ,45 ,45 ,45 ,
213 ,63 ,63 ,63 ,55 ,45 ,109 ,4
5 ,213 ,63 ,31 ,63 ,119 ,73 ,41
 ,173 ,59 ,223 ,219 ,74 ,73 ,9
,213 ,223 ,219 ,19 ,0
2060  DATA  0 ,8 ,109 ,73 ,209
 ,255 ,31 ,191 ,77 ,45 ,45 ,213
 ,27 ,63 ,63 ,119 ,73 ,45 ,173
,219 ,219 ,155 ,109 ,73 ,137 ,2
19 ,27 ,63 ,55 ,0 ,0 ,0
2070  DATA  0 ,40 ,45 ,45 ,77
,218 ,63 ,63 ,31 ,110 ,109 ,109
 ,26 ,63 ,255 ,31 ,110 ,41 ,45
,173 ,27 ,63 ,63 ,63 ,46 ,45 ,4
5 ,109 ,218 ,59 ,223 ,55
2080  DATA  0 ,40 ,109 ,109 ,2
09 ,219 ,31 ,63 ,46 ,109 ,73 ,2
09 ,219 ,219 ,51 ,77 ,73 ,137 ,
219 ,219 ,155 ,9 ,77 ,73 ,218 ,
219 ,59 ,55 ,0 ,0 ,0 ,0
2090  DATA  0 ,72 ,105 ,73 ,21
8 ,223 ,251 ,10 ,77 ,109 ,209 ,
223 ,251 ,119 ,77 ,109 ,209 ,25
1 ,27 ,159 ,73 ,77 ,137 ,219 ,2
19 ,155 ,0 ,0 ,0 ,0 ,0 ,0
2100  DATA  0 ,72 ,9 ,77 ,209
,27 ,223 ,187 ,9 ,109 ,105 ,26
,255 ,223 ,115 ,41 ,77 ,141 ,21
9 ,223 ,187 ,73 ,105 ,137 ,219
,219 ,155 ,0 ,0 ,0 ,0 ,0
2110  DATA  0 ,72 ,73 ,109 ,21
8 ,255 ,31 ,55 ,45 ,45 ,77 ,213
 ,63 ,63 ,255 ,42 ,45 ,77 ,137
,219 ,219 ,155 ,73 ,73 ,173 ,59
 ,63 ,223 ,19 ,0 ,0 ,0
2120  DATA  0 ,72 ,45 ,45 ,173
 ,251 ,63 ,63 ,87 ,109 ,109 ,21
3 ,31 ,31 ,63 ,55 ,45 ,45 ,77 ,
213 ,63 ,63 ,63 ,87 ,45 ,45 ,45
 ,213 ,255 ,59 ,159 ,0
2130  DATA  0 ,72 ,109 ,45 ,21
3 ,63 ,31 ,223 ,74 ,73 ,45 ,213
 ,255 ,219 ,83 ,73 ,73 ,213 ,21
9 ,219 ,83 ,73 ,105 ,209 ,63 ,2
23 ,155 ,0 ,0 ,0 ,0 ,0
2140  DATA  0 ,8 ,109 ,73 ,209
 ,219 ,219 ,23 ,109 ,73 ,137 ,2
19 ,219 ,63 ,46 ,109 ,73 ,209 ,
219 ,219 ,55 ,109 ,9 ,77 ,218 ,
59 ,223 ,55 ,0 ,0 ,0 ,0
2150  DATA  0 ,40 ,45 ,13 ,77
,218 ,251 ,27 ,55 ,109 ,73 ,141
 ,59 ,63 ,63 ,63 ,110 ,73 ,73 ,
218 ,219 ,27 ,55 ,45 ,77 ,73 ,2
18 ,219 ,63 ,55 ,0 ,0
2160  DATA  0 ,40 ,45 ,45 ,109
 ,26 ,63 ,63 ,63 ,55 ,45 ,13 ,
 45 ,173 ,59 ,63 ,31 ,63 ,46 ,
 109 ,73 ,209 ,219 ,219 ,55 ,
 77 ,73 ,137 ,219 ,219 ,27 ,6
2170  DATA  0 ,72 ,73 ,73 ,218
 ,219 ,219 ,74 ,73,73,218 ,22
3 ,219 ,74 ,73 ,109 ,218 ,63 ,2
23 ,83 ,73 ,41 ,141 ,27 ,31 ,22
3 ,19 ,0 ,0 ,0 ,0 ,0
2180  DATA  0 ,72 ,45 ,77 ,
209 ,219 ,59 ,63 ,46 ,77
,73 ,209  ,219 ,219 ,83
,73 ,73 ,209 ,219
,219 ,83 ,73 ,73
,209 ,219 ,21
9 ,19 ,0 ,0
,0 ,0 ,0
```

ANIMAÇÃO GRÁFICA NO TRS-COLOR

Com os comandos GET e PUT podemos armazenar figuras em matrizes e, mais tarde, trazê-las de volta à tela. Aprendendo a utilizá-los, você animará os mais variados desenhos.

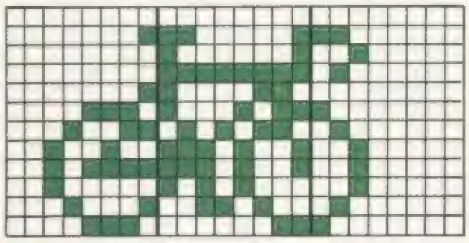
Se você já sabe elaborar gráficos com UDG (Caracteres Definidos pelo Usuário), é quase certo que queira movê-los pela tela. Para tanto, será necessário utilizar os comandos **GET** e **PUT**, que já vimos muitas vezes na seção *Programação de Jogos*.

Poderíamos imaginar o **GET** como um comando capaz de "fotografar" uma área retangular da tela de alta resolução do TRS-Color. O que estiver naquela área — um UDG ou qualquer outro gráfico feito com **LINE**, **CIRCLE** ou **DRAW** — será armazenado numa matriz (veja artigo na página 192).

PUT é o oposto de **GET**: ele "coloca" a fotografia (conteúdo da matriz) em qualquer lugar da tela. Usar esse comando é muito mais rápido que redesenhar o gráfico por outros métodos e, com ele, a idéia de movimento pode ser facilmente simulada: basta colocar (**PUT**) uma imagem em diferentes posições na tela ou, então, colocar imagens diferentes repetidamente.

BICICLETA

Digite e rode este programa; ele usa **GET** e **PUT** para movimentar um UDG de uma bicicleta.



```
10 PMODE 4,1:PCLS
20 DIM BY(5)
30 FOR K=1536 TO 1856 STEP 32
40 READ A,B,C
50 POKE K,A:POKE K+1,B:POKE K+2
,C
60 NEXT
70 SCREEN 1,1
80 GET (1,0)-(18,11),BY,G
90 PCLS
100 FOR K=0 TO 238
110 PUT (K,90)-(K+17,100),BY,PS
ET
120 NEXT
130 GOTO 90
140 DATA 1,193,192,0,129,32,0,2
55,64,1,134,0,14,139,128,19,86,
64
150 DATA 36,233,32,47,233,32,32
,104,32,17,100,64,14,3,128
```

Os **DATA** (dados) das linhas 140 e 150 são acessados e depois inseridos na tela, via comando POKE, nas linhas 30 a 60. A bicicleta é desenhada no canto superior esquerdo, ou seja, no começo da tela. Quando você for inserir um grá-

- ANIMAÇÃO DE UM CARACTERE DEFINIDO PELO USUÁRIO (UDG)
- O USO DE MATRIZES COM OS COMANDOS **GET** E **PUT**
- "FOTOGRAFE" A TELA
- COMO DIMENSIONAR CORRETAMENTE UMA MATRIZ
- COLOQUE O GRÁFICO ONDE QUISER
- TÉCNICAS DE ANIMAÇÃO

fico na tela e depois usar **GET** e **PUT**, desenhe-o nessa posição, pois assim saberá exatamente onde ele está. Converter posições de memória em posições de tela pode não ser fácil, devido a variações entre os **PMODE**.

A linha 80 faz um **GET** (ou "tira uma fotografia") da bicicleta. A linha 90 limpa a tela antes que as linhas 100 e 120 movimentem a bicicleta. A linha 110 **PUT** ("coloca") a bicicleta em uma posição diferente cada vez que passa pelo laço **FOR...NEXT**.

COMO DIMENSIONAR UMA MATRIZ

Sempre que usar **GET** num programa, você precisará dimensionar **(DIM)** uma matriz para armazenar as informações sobre o gráfico. Para calcular o tamanho da matriz faça o seguinte:

1) Depois de desenhar o gráfico, de preferência em papel quadriculado, trace um retângulo, "enquadrando" inteiramente a figura. Conte quantos quadrados existem na primeira linha e quantos existem num dos lados. Multiplique estes dois números — o resultado será o número de quadrados que o desenho ocupa. Por exemplo: o retângulo que envolve a bicicleta tem dezoito quadrados de largura por doze quadrados de altura. Multiplicando-os, obtém-se o número total de quadrados, 216.

2) Em seguida, calcule o número de bytes de memória necessário para armazenar todos esses quadrados. Em **PMODE** 3 e 4 divide-se o número de quadrados por 8, em **PMODE** 1 e 2 divide-se por 16, e em **PMODE** 0 por 32. Como a bicicleta foi desenhada em **PMODE** 4, divide-se 216 por 8, obtendo-se 27. O resultado deve ser um número inteiro de bytes e, por isso, às vezes é preciso ajustá-lo. E, seja qual for o número obtido, temos sempre que arredondá-lo para cima, nunca para baixo.

3) Para calcular o tamanho que deve ter a matriz, divida o número de bytes por 5 (o divisor é sempre 5, não importa o **PMODE** que está sendo usado).

Voltando ao exemplo da bicicleta: temos 27 bytes para armazenar na matriz; portanto, 27/5 = 5,4. Novamente, se o resultado não for inteiro, deve-se arredondá-lo para cima — 6, nesse caso. A linha 20 diz **DIM BY (5)** porque as matrizes começam em 0.

Aqui está um resumo dos divisores necessários para calcular o tamanho das matrizes nos programas gráficos:

Para calcular o número de:

PMODE	Bytes	Matrizes
4	8	5
3	8	5
2	16	5
1	16	5
0	32	5

Pode-se usar **GET** com todo tipo de gráfico na tela de alta resolução do computador. Quer utilizemos um UDG, os comandos **PSET** e **PRESET**, **DRAW**, ou qualquer combinação de comandos gráficos, será sempre possível armazenar figuras em matrizes por meio de **GET** e colocá-las **(PUT)** de volta à tela mais tarde.

Para armazenar um gráfico numa matriz previamente dimensionada, você precisará dizer ao computador onde encontrar o gráfico na tela e em qual matriz pretende armazená-lo. Vejamos esta linha do programa da bicicleta:

```
80 GET(1,0)-(18,11),BY,G
```

Os números nos parênteses são as coordenadas de cantos diagonalmente opostos do retângulo que envolve o gráfico. Você pode especificar os cantos que quiser e em qualquer ordem, desde que sejam diagonalmente opostos.

A área armazenada não compreende todo o UDG que definimos, mas apenas as partes que realmente contêm a imagem — porque a bicicleta não ocupa toda a extensão de 24 quadrados. Como o **GET** não nos limita a mover apenas imagens de oito por oito, podemos movimentar o que bem entendermos.

Na continuação da linha 80, **BY** manda o computador armazenar o gráfico na matriz **BY**, **DIM**ensionada na linha 20.

G significa "detalhe Gráfico completo". Pode-se omiti-lo em certos casos (que explicaremos depois), mas em geral é aconselhável usá-lo.

Se estiver usando UDG, convém sempre inseri-los **(POKE)** no canto superior esquerdo da tela, como foi dito anteriormente, para facilitar o ajuste do **GET**. Lembre-se de que quando os inserimos na tela a posição da imagem não está contida em coordenadas comuns. O **GET** "fotografa" a imagem de uma certa coordenada de tela e, por isso, será preciso converter a posição de memória na qual introduzimos os UDG em coor-

denadas de tela equivalentes. Essa conversão não é fácil, já que varia conforme o **PMODE**. A inserção nas posições de memória do começo da tela (a partir da posição 1536) facilita o cálculo da coordenada de tela do UDG. O canto superior esquerdo deve estar na coordenada (0,0) e, sabendo o tamanho do gráfico, será simples calcular as coordenadas do canto inferior direito do UDG quando o "fotografarmos".

O USO DO PUT

Depois de empregarmos o **GET** para armazenar o gráfico na memória, devemos limpar a tela com **PCLS**, antes de colocá-lo **(PUT)** de volta na tela.

Aqui está a linha do **PUT** no programa da bicicleta:

```
110     PUT(K,90)-(K+17,100),BY,
PSET
```

Dentro dos parênteses, como no **GET**, encontram-se as coordenadas dos cantos diagonalmente opostos que especificam a área na qual queremos colocar a imagem. **BY** é a matriz que contém a bicicleta.

A última parte da linha é o **PSET**, que faz o computador colocar o gráfico na tela exatamente como foi desenhado, sobrepondo-o ao que estiver naquelas posições.

Existem, porém, outras opções. **PSET** pode ser substituído por **PRESET**, **OR**, **AND**, ou **NOT**, que nos permitem manipular o gráfico.

PRESET diz ao computador para colocar o gráfico na tela no modo inverso. No modo de duas cores, estas se invertem; no de quatro (modo 0), o vermelho fica verde, o azul fica amarelo e vice-versa; no modo 1, o laranja fica cinza, o ciano fica roxo e vice-versa.

OR permite que se coloque o gráfico armazenado na matriz junto ao que já está na tela. Ambos permanecem inalterados, o que não ocorre quando utilizamos **PSET**, que elimina o que estiver na tela.

AND mostra a junção do que já está na tela com o gráfico que se introduz naquela posição.

NOT não mostra nenhum conteúdo da matriz: simplesmente inverte a área da tela em que a matriz será colocada.

A seguir, você verá a utilização desses comandos em um programa.

```
10 PMODE 4,1
20 DIM C(5),B1(5),B2(5)
30 PCLS
40 CIRCLE (7,7),7,1:PAINT (7,7)
,1,1
50 GET (0,0)-(14,14),C,G
60 PUT (0,0)-(14,14),C,NOT
70 GET (0,0)-(14,14),B2,G
80 PCLS1:SCREEN 1,0
90 LINE(8,191)-(104,0),PRESET:L
INE(24,191)-(120,0),PRESET:PAIN
T(10,191),0,0
100 LINE (104,191)-(200,0),PRES
ET:LINE(120,191)-(216,0),PRESET
:PAINT(110,191),0,0
110 FOR K=175 TO 0 STEP -10
120 X=14+(175-K)/2
130 PUT (X,K)-(X+14,K+14),C,OR
140 PUT(X+96,K)-(X+110,K+14),C,
OR
150 FOR J=1 TO 100:NEXT
160 PUT(X,K)-(X+14,K+14),B1,PSE
T
170 PUT(X+96,K)-(X+110,K+14),B2
,AND
180 NEXT
190 GOTO 80
```

Se observarmos bem as duas bolas subindo pela tela, veremos que sobram "pontas" ao longo das faixas pretas. Isso acontece porque estamos tentando colocar um gráfico retangular dentro de um inclinado e, cada vez que o programa introduz um quadrado preto (lacuna) para "apagar" a posição anterior da bola, as "pontas" aparecem.

Com a bola que caminha pela faixa da direita não ocorre esse problema. Usando uma combinação de **PUT...**, **AND** e **PUT...,OR**, as pontas são eliminadas. A linha 60 inverte a bola na tela e, depois, a linha 70 armazena-a na memória. Quando o programa movimenta a bola, a linha 140 coloca uma lacuna sobre a faixa com um **PUT...,OR**, de maneira que a bola anterior seja apagada por uma "figura" de formato exatamente igual ao dela. Em seguida, um **PUT...,AND** coloca na tela a bola invertida. A combinação de **NOT** com **AND** permite que a linha 170 introduza a bola na tela sem superposição.

Se você quiser colocar um gráfico na tela, num espaço que não seja retangular (um círculo num triângulo, por exemplo), desenhe primeiro o objeto. Coloque-o de volta à tela com **PUT...,NOT** e "fotografe-o" **(GET)** numa matriz. Mais tarde, para mostrar o gráfico na tela, use **PUT...,AND**.

QUANDO NÃO USAR O G

Dissemos anteriormente que, em certos casos, pode-se omitir o **G** do final do comando **GET**. Para saber quando omiti-lo, você precisa de algumas informações sobre a tela do computador.

Nos **PMODE** 1, 3 e 4 a tela gráfica é dividida em 32 colunas verticais, cada uma com oito pontos de largura, enquanto que nos **PMODE** 0 e 2 a tela é dividida em dezesseis colunas de dezesseis pontos de largura cada. Qualquer que seja o **PMODE** usado, a largura de cada coluna na tela corresponde a um byte na memória.

Pode-se omitir o **G** do comando **GET** quando o gráfico desenhado ocupa um número determinado de colunas e se quer movimentá-lo. Sem o **G**, bytes inteiros são armazenados na matriz declarada. Se o gráfico sobrepuser parte de colunas, você precisará armazenar partes de bytes na matriz. Em outras palavras, será preciso, de alguma maneira, armazenar bits da tela na matriz — e é esta a função do **G**.

Retirando o **G**, você terá problemas com gráficos que sobrepõem outras colunas. Além disso, não poderá usar **PSET**, **PRESET**, **OR**, **AND** ou **NOT** no **PUT** correspondente. A omissão do **G**, portanto, torna menos flexível o uso do **PUT**.

ANIMAÇÃO COM GET E PUT

O **GET** e o **PUT** são especialmente úteis nos programas de animação gráfica. Esta é feita, em geral, por meio de dois métodos.

O primeiro deles consiste em usar uma matriz vazia (lacuna) para apagar o gráfico, antes de colocá-lo em outra posição na tela. Para evitar problemas, verifique antes se o tamanho da lacuna é o mesmo do gráfico. Esse procedimento já foi visto em vários artigos de *Programação de Jogos*.

O segundo método, empregado na animação da bicicleta, utiliza uma moldura de pontos vazios ao redor do gráfico. Depois que se decide quantos pontos o gráfico se moverá a cada passo, verifica-se se existe o mesmo número de fileiras de pontos rodeando o gráfico. Isso significa que o novo gráfico colocado na tela apaga o anterior. Se o gráfico se movesse para todas as direções, quatro pontos a cada passo, por exemplo, precisaríamos de uma moldura com quatro pontos de largura rodeando-o. No caso da bicicleta, o gráfico se move um ponto de cada vez, da esquerda para a direita somente. Assim, quando ele foi armazenado, deixamos uma simples fileira de pontos vazios em seu lado esquerdo.

Como exercício, tente animar outros desenhos usando **GET** e **PUT**. A única maneira de entender e se familiarizar com o uso dos comandos **PSET**, **PRESET**, **OR**, **AND** e **NOT** é experimentá-los em diversos gráficos e situações.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
O TK-2000 não dispõe da função **INKEY$**. Veja como substituí-la.

APLICAÇÕES
Planeje UDG sem trabalho: deixe os cálculos para o computador.

PROGRAMAÇÃO BASIC
Os gráficos lineares são muito úteis à organização de informações numéricas. Aprenda a programá-los.

PROGRAMAÇÃO BASIC
Conheça o **PRINT USING**, poderoso comando auxiliar do **PRINT**.

PERIFÉRICOS
Fitas e discos danificados podem produzir efeitos desastrosos. Cuide deles.